

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

FINJAN SOFTWARE, LTD., an Israel
corporation,

Plaintiff,

v.

SECURE COMPUTING CORPORATION,
a Delaware corporation, CYBERGUARD,
CORPORATION, a Delaware corporation,
WEBWASHER AG, a German corporation
and DOES 1 THROUGH 100,

Defendants.

C. A. No. 06-369-GMS

PUBLIC VERSION

**AFFIDAVIT OF MEGHAN ASHLEY WHARTON IN SUPPORT OF
FINJAN'S MOTION *IN LIMINE* NO. 1 TO EXCLUDE TESTIMONY OF
DAN WALLACH'S INVALIDITY OPINION REGARDING FINJAN'S
THREE PATENTS IN SUIT**

OF COUNSEL

Paul J. Andre
Lisa Kobialka
Perkins Coie LLP
101 Jefferson Drive
Menlo Park, CA 94025-1114
(650) 838-4300

Philip A. Rovner (#3215)
POTTER ANDERSON & CORROON LLP
Hercules Plaza
P. O. Box 951
Wilmington, DE 19899
(302) 984-6000
provner@potteranderson.com

Attorneys for Plaintiff
Finjan Software, Ltd.

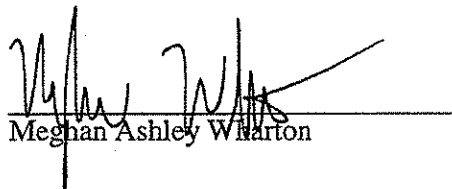
Dated: December 27, 2007
Public Version: January 14, 2008

I, MEGHAN ASHLEY WHARTON, declare:

1. I am an attorney with the law firm of Perkins Coie LLP, counsel of record for Finjan Software, Ltd ("Finjan"). I have personal knowledge of the facts set forth in this affidavit and can testify competently to those facts.
2. Attached hereto as Exhibit 1 is a true and correct copy of Opening Technical Expert Report of Dan Wallach ("Wallach Opening Report"), served on November 12, 2007.
3. Attached hereto as Exhibit 2 is a true and correct copy of letter from Trevor Foster, counsel for Secure Computing Corp. ("Secure Computing"), to James Hannah, counsel for Finjan, dated November 30, 2007, enclosing supplemental exhibits C thorough J to the Wallach Opening Report.
4. Attached hereto as Exhibit 3 is a true and correct copy of pages 14-15, 40-41 and 134 from the rough transcript of the deposition of Dan Wallach.

I declare under penalty of perjury under the laws of the State of California and the United States that each of the above statements is true and correct.

Executed on December 27, 2007, in Menlo Park, California.



Meghan Ashley Wharton

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

CERTIFICATE OF SERVICE

I, Philip A. Rovner, hereby certify that on January 14, 2008, the within document was filed with the Clerk of the Court using CM/ECF which will send notification of such filing(s) to the following; that the document was served on the following counsel as indicated; and that the document is available for viewing and downloading from CM/ECF.

BY HAND DELIVERY AND E-MAIL

Frederick L. Cottrell, III, Esq.
Kelly E. Farnan, Esq.
Richards, Layton & Finger, P.A.
One Rodney Square
920 N. King Street
Wilmington, DE 19801
cottrell@rlf.com; farnan@rlf.com

I hereby certify that on January 14, 2008 I have sent by E-mail the foregoing document to the following non-registered participants:

Jake M. Holdreith, Esq.
Christopher A. Seidl, Esq.
Robins, Kaplan, Miller & Ciresi L.L.P.
2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, MN 55402
jmholdreith@rkmc.com ; caseidl@rkmc.com

/s/ Philip A. Rovner
Philip A. Rovner (#3215)
Potter Anderson & Corroon LLP
Hercules Plaza
P.O. Box 951
Wilmington, Delaware 19899
(302) 984-6000
E-mail: provner@potteranderson.com

EXHIBIT 1, PART 1

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

FINJAN SOFTWARE, LTD., an Israel
corporation,

Plaintiff,

v.

SECURE COMPUTING CORPORATION, a
Delaware corporation; CYBERGUARD
CORPORATION, a Delaware corporation,
WEBWASHER AG, a German corporation and
DOES 1 THROUGH 100,

Defendants.

Civil Action No. 06-00369 GMS

OPENING TECHNICAL EXPERT REPORT OF DAN WALLACH

I. INTRODUCTION

1. I have been retained as an expert witness in this action by Defendant-Counterclaim Plaintiff, Secure Computing Corporation. I will testify on whether certain prior art anticipates or otherwise renders obvious the claims asserted by Plaintiff-Counterclaim Defendant, Finjan Software, Ltd. in U.S. Patent Nos. 6,092,194 (hereafter, the '194 patent), 6,804,780 (hereafter, the '780 patent), and 7,058,822 (hereafter, the '822 patent). I will also testify as to whether Finjan's Vital Security NG appliances and Internet 1Box commercial products from Finjan Software, Ltd. and Finjan Software, Inc., infringe certain claims asserted by Secure Computing Corp. from U.S. Patent No. 7,185,361 (hereafter, the '361 patent). I will further testify as to whether Finjan Mirage (aka Vital Security for Documents or Vital Security for Enterprise Documents) infringes certain claims asserted by Secure Computing Corp. from U.S. Patent No. 6,357,010 (hereafter, the '010 patent).

2. This report reflects my opinions with respect to the claims that counsel has asked me to evaluate based on the information made available to me. I reserve the right to supplement or amend this report based on information I receive after the date of this report.

II. QUALIFICATIONS

3. I am an Associate Professor in the Department of Computer Science at Rice University in Houston, Texas. I earned my doctorate degree at Princeton University in 1999 and my master of arts at Princeton in 1995. I earned my bachelor's of science in electrical engineering and computer science at the University of California, Berkeley in 1993. My specialty is computer security.
4. I began focusing on computer security during my doctoral studies at Princeton in 1995 when I began my investigations into the security of Sun's Java technologies. My studies included examining how Java worked, both by reading the source code and constructing attacks against the system. Working with my Princeton collaborators, Drew Dean, Ed Felten, and Dirk Balfanz, we published several papers over the next several years concerning security issues in Java. We also gave a number of talks and presentations. A list of my publications and a representative sample of my presentations can be found in my curriculum vitae (see Exhibit A).
5. I have studied, more generally, the security of web browsers and web servers. In addition to finding a number of security flaws, I helped Netscape improve the security of its Java system while I was a summer intern there in 1996 and 1997. In the course of my academic studies, I have authored or co-authored over forty refereed papers on these and other topics in computer security.
6. After receiving my doctorate, I accepted the position of assistant professor at Rice

University in the Department of Computer Science. I am currently an associate professor with tenure. While at Rice, the focus of my research has been on a number of different topics in computer security, including mobile code security and the security of peer-to-peer networking systems. My research group has been supported by various corporations and grant agencies: Microsoft, Schlumberger, IBM, the National Science Foundation, and the Texas Advanced Technology Program.

7. I have served on a number of program committees for research conferences in computer security; these positions are generally considered to be a recognition of expertise and achievement. I was also honored to serve as the program chair for the USENIX Security Symposium in 2001 and the track co-chair for the International World Wide Web Conference's Security and Privacy Track in 2007 and 2008.
8. A complete list of my publications, program committee appointments, testimony, grants received and other honors appears in my curriculum vita, attached as Exhibit A.
9. I am being compensated at the rate of \$450/hour. This compensation does not depend on the outcome of this case.

III. INFORMATION CONSIDERED

10. To prepare this report I reviewed Secure Computing and Finjan Patents in this lawsuit and reviewed and/or received certain documentation and materials as shown in Exhibit B to this report. My opinions are based on my education, background, training, and experience, and on the reviews of the aforementioned documents.
11. I understand from counsel that there may be more information that has not yet been made available to Secure Computing. If I am given access to this information, I will supplement my opinions at a later date.

IV. TERMS

12. In this opinion, I use the ordinary meaning of all terms in the patent claims for which there is an ordinary meaning, except that I am aware of the specific definitions of claim terms proposed by Secure Computing and by Finjan for some terms. In many instances, my opinions are not affected by which construction the court may order for the disputed terms because my analysis is true for each case. Where the court's claim construction may make a difference to my opinion, I have attempted to point out the difference. I may supplement my opinion once the court issues its claim construction

V. APPROPRIATE PRIORITY DATES

13. Counsel has told me that the priority date of the patent is used to assess whether or not a particular reference or article may appropriately be considered prior art. Counsel has further informed me that in order for a non-provisional application to claim the benefit of the earlier filing date of a provisional application "the specification of the provisional 'must contain a written description of the invention and the manner and process of making and using it, in such full, clear, concise, and exact terms,' 35 U.S.C. § 112 P 1, to enable an ordinarily skilled artisan to practice the invention claimed in the non-provisional application." *New Railhead Mfg., L.L.C. v. Vermeer Mfg. Co.*, 298 F.3d 1290, 1294 (Fed. Cir. 2002) (emphasis in original).

Priority Date of the '194 Patent

14. Counsel has informed me that a claim in a non-provisional application may not claim the invention of a broader subject matter than was disclosed in the provisional application if the patentee intends to benefit from the earlier filing date of the earlier filing date of the provisional application. While I offer no opinion or interpretation regarding the legal

aspects of this standard, I have been asked to evaluate whether the provisional application dated November 8, 1996, describes what is in the issued claims of the '194 patent and the manner and process of making and using it in clear, concise, and exact terms to enable an ordinarily skilled artisan to practice the invention claimed in the '194 patent.

15. The provisional application dated Nov. 8, 1996 by the patent office, states "A Downloadable is an executable application program which is *automatically* downloaded from a source computer and run on the destination computer." (FIN 011072) (emphasis added) Contrasting, the issued patent states "A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer." ('194 1:44-47) It is my opinion that the "executable application program, which is downloaded" of the '194 patent's non-provisional application is a different, and broader, subject matter than the "automatically downloaded" executable that is disclosed in the provisional application. The "downloadable" was broadened by eliminating the word "automatically" from the previous definition. This new definition is thus broader because it includes programs which are downloaded manually. Counsel has informed me that if the Court finds that the "downloadable" described in the earlier application is different from the "downloadable" in the later application, the priority date for the '194 patent is the same as the filing date of the later, non-provisional application: November 6, 1997. Because it is my opinion that the terms are different in the two applications, I rely on the later priority date of November 6, 1997, the date of the application that makes clear that the inventors believe "A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer" as also stated in the claims of the "194 patent.

Priority Date of the '780 Patent

16. As with the '194 patent, every claim of the '780 patent includes the limitation of "Downloadable." And, as with the '194 patent, the definition of "Downloadable" is narrower in the provisional application.
17. Moreover, each independent claim of the '780 patent includes the limitation "performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID." Every claim includes "references to software components required to be executed by the Downloadable." These "software components" are plainly distinct from "Downloadables", but the provisional application has no discussion of "software components." The provisional application offers no assistance in understanding what a "referenced software component" is, much less enables one of ordinary skill in the art how to fetch those referenced software components and perform a hashing function on the referenced software components and the Downloadable. Thus, the provisional application does not describe hashing referenced software components as claimed.

Priority Date of the '822 Patent

18. The provisional application for the '822 patent was filed on May 17, 2000, while the final application was filed on May 17, 2001. The final application's claims includes a variety of non-standard terms that are entirely absent from the provisional application, including downloadable-information, mobile protection code, detection-indicators, downloadable-information characteristics, executable code characteristics, information-destinations, downloadable-information analyzers, inspection controller, information monitor, content inspection engine, packaging engine, linking engine, and information re-communicator. It is my opinion that, absent some definitions or discussion of the non-standard terms, one

of ordinary skill in the art would not be enabled to practice the invention claimed in the non-provisional application based on the provisional application.

VI. SUMMARY OF OPINIONS RELATED TO INVALIDITY

19. Counsel informed me that Finjan has asserted the following claims:

- '194 Patent--Claims 1-14, 24-30, 32-36, and 65;
- '780 Patent--Claims 1-18; and
- '822 Patent--Claims 1, 2, 4, 6, 8, 9, 12, 13, 15-18, 20-22, 24, 26-29, 31, 32, 34, and 35.

20. This particular report will only address certain prior art that, in my opinion, anticipates or otherwise renders obvious Finjan's asserted claims. I understand the Finjan has not yet disclosed its contentions as to the validity of the Finjan patents over the prior art. I reserve the right to amend the opinions expressed in this report based on any additional information that I may receive.

21. For convenience of the Court, in the body of this report, I have summarized my opinions with respect to the prior art and any other grounds for invalidity. In this summary, I have discussed all of the asserted independent claims in Finjan's patents. A detailed basis for the analysis is attached to this report as Exhibits C-J. All asserted claims, both independent and dependent, are examined in these exhibits.

22. In my opinion, all of the asserted claims in the '194 Patent are anticipated or otherwise rendered obvious by:

- Shaio et al (U.S. Patent No. 6,571,338) (filed Dec. 20, 1995) (hereafter "**Shaio**"); and
- The Firewall Toolkit (**FWTK**) 2.0 Beta release (Revised March 1996 - Sept. 1996) (hereafter "**FWTK**").

23. In my opinion, all of the asserted claims in the '194 Patent are rendered obvious by the

following combination of references:

- Hershey et al (U.S. Patent No. 5,414,833) (hereafter "**Hershey**") in combination with Raymond W. Lo. et. al., *Towards a Testbed for Malicious Code Detection*, Compcon Spring '91, Digest of Papers (25 Feb.-1 Mar. 1991) at 160-66, available at http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=128800 (hereafter "**Lo 1991**");
- Hershey in combination with Raymond W. Lo. et. al., MCF: A Malicious Code Filter (May 4, 1994), available at <http://seclab.cs.ucdavis.edu/papers/llo95.ps> (hereafter "**Lo 1994**");
- Ji et al (U.S. Patent No. 5,623,600) (filed Sept. 26, 1995) (hereafter "**Ji 1995**") in combination with Lo 1991;
- **Ji 1995** in combination with **Lo 1994**;
- Chen et al (U.S. Patent No. 5,951,698) (filed Oct. 2, 1996) (hereafter "**Chen**") in combination with Hershey; and
- **Chen** in combination with **Ji 1995**.

24. The preceding combinations are each based on a combination of two references. It is not necessary to combine more than two of the above references to render obvious the asserted claims of the '194 Patent.
25. In my opinion, all of the asserted claims in the '780 Patent are anticipated or otherwise rendered obvious by knowledge of one of ordinary skill in the art in combination with known methods of applying digital signature to code, such as Microsoft Authenticode. (Microsoft Corp., Microsoft Authenticode Technology: Ensuring Accountability and Authenticity for Software Components on the Internet (Oct. 1996) (hereafter "**Authenticode**"))
26. In my opinion, all of the asserted claims in the '822 Patent are anticipated or otherwise rendered obvious by Ji, U.S. Patent No. 5,983,348 (filed Sept. 10, 1997) (hereafter "**Ji 1997**").

VII. LEVEL OF ORDINARY SKILL IN THE ART

27. I have been asked by counsel to assume that a person of ordinary skill in the art would have a computer science, computer engineering or electrical engineering degree with at least five (5) years of experience in the computer industry including specific experience in the field of computer security. I have not attempted to make my own investigation of the level of skill in the art. I have at least this level of skill in the art based on my 12+ years in the area of computer security.

VIII. LEGAL STANDARDS FOR INVALIDITY

28. Counsel has instructed me to rely on the following legal standards with respect to anticipation and obviousness.
29. Counsel has instructed me that under 35 U.S.C. § 282, a patent is presumed valid, and that the party challenging validity has the burden of proving invalidity by clear and convincing evidence. Counsel has instructed me that anticipation under 35 U.S.C. § 102 requires that a prior art reference must disclose each and every element of a claim, either expressly or inherently.
30. Counsel has instructed me that obviousness under 35 U.S.C. § 103 considers whether the differences between the claims and the prior art are such that the claims would have been obvious at the time of invention to one of ordinary skill in the art. Counsel has instructed me that the U.S. Supreme Court recently interpreted this standard in *KSR Int'l Co. v. Teleflex, Inc.* *KSR* explained that under the so-called *Graham* factors, the following must be considered: (1) the scope and content of the prior art; (2) the differences between the art and the claims at issue; (3) the level of ordinary skill in the art; and (4) objective evidence of nonobviousness. Counsel has instructed me that, as a part of my analysis, I

am to consider whether the differences between the prior art and the claims at issue are substantial or insubstantial.

31. Counsel has instructed me that under both § 102 and § 103, I must compare each and every element of the asserted claims to the prior art.

IX. SCOPE OF TESTIMONY

32. I will offer testimony at trial regarding my opinions expressed herein and within the attachments. In addition, I will offer rebuttal testimony as appropriate. I may offer testimony by way of background regarding how computer networks function in general, how the Internet functions in general, and other relevant technological background, such as the background of executable code, computer attacks, and computer security.

X. THE '194 PATENT IS INVALID FOR REASONS OF ANTICIPATION AND OBVIOUSNESS

Summary of the '194 Patent

33. The '194 patent pertains to the field of computer and network security. The '194 patent generally describes a process for protecting end users against the risks associated with potentially hostile executable code, including, e.g., Java applets, ActiveX controls, and JavaScript scripts (referred to in the patent as "Downloadables"), which can accompany web pages, emails, and so forth. One traditional method of blocking potentially hostile executable code (e.g., computer viruses) is called "signature scanning". This process compares requested Downloadables against a collection of previously known "signatures" of known-hostile code, and removes them if the signature matches to prevent the user from being attacked. The '194 patent considers the case of Downloadables which aren't already known to be hostile and for which no known signatures already exist to detect them. A specific method using a list of suspicious computer operations is then described

to statically scan these Downloadables for potentially hostile behavior, which if detected would trigger the Downloadable being removed, i.e., not transmitted to the client. All of the scanning and/or filtering, as described in the patent, would take place on a server that intercepts web or other network traffic addressed to the client before it reaches the client.

State of the Art

34. As early as the 1980s and continuing today, signature scanning is a widespread technique for detecting viruses or other hostile code. Technicians working at anti-virus companies would examine infected systems (or infected floppy disks) and would hand-craft signatures, i.e., patterns in the program files which can be detected without needing to execute the program. Customers of anti-virus companies would then receive regular updates including the newest anti-virus patterns.
35. Although signature-based techniques were an effective way of preventing known viruses, they were of little use against previously unknown viruses or other unknown hostile code. Alternative approaches were developed both in industry and academia that could detect new and previously unknown hostile code.
36. At least two techniques, known as static and dynamic methods, were used as early as 1991 to detect such unknown attacks. As one 1991 article describes, "Unlike most viruses detection techniques, two types of analysis attempt to peer inside a program to detect what it is doing and how. Static analysis methods can determine certain properties for some types of programs. Dynamic analysis methods attempt to learn more about a program's behavior by actually running it or by simulating its execution."¹
37. In the early 90s static analysis to detect unknown attacks was focused on analyzing the

¹ Raymond W. Lo. et. al., *Towards a Testbed for Malicious Code Detection*, Compcon Spring '91, Digest of Papers (25 Feb.-1 Mar. 1991) at 162.

code for suspicious operations or behavior that the code might attempt. One 1993 article summarizes:

The heuristic scanners that I am familiar with are able to detect suspicious instruction sequences, like the ability to format a disk, the ability to search for other executables, the ability to remain resident in memory, the ability to issue non-standard or undocumented system calls, etc. Each of these abilities has a value assigned to it. If the total of the values for one program exceeds a pre-defined threshold, the scanner yells 'Virus!'. A single suspected ability is never enough to trigger the alarm. It is always the combination of the suspected abilities which convince the scanner that the file is a virus."²

38. While new techniques for preventing hostile attacks were developing, so was the popularity of computer networks. As early as 1993, techniques that were previously used solely on individual computers began migrating to use within computer networks.³

Comparison of the Prior Art to the Asserted Claims of the '194 Patent

39. As discussed above, for the convenience of the court, I am providing a summary of my opinion with respect to the sole asserted independent claim of the '194 patent. For a detailed element-by-element analysis of my opinions please refer to Exhibits C-J.
40. The sole asserted independent claim of the '194 patent is claim 1. Below is a summary of claim 1 and a list of reasons why I believe claim 1 to be invalid.
41. The '194 patent claim 1 recites:

A computer-based method, comprising the steps of:
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security

² Franz Veldman, Combating Viruses Heuristically, Virus Bulletin Conference, Sept. 1993, at 68. See also W. Timothy Polk et al, Anti-Virus Tools and Techniques For Computer Systems 24 (1995) (indicating that static analysis detection tools that analyze code looking for "suspicious, virus-like behavior . . . have been used successfully to identify executables infected by 'new' viruses in a few actual outbreaks.")

³ See Ji et al (U.S. Patent No. 5,623,600); Hershey et al (U.S. Patent No. 5,414,833)

policy to determine if the security policy has been violated; and *preventing* execution of the Downloadable by the client if the security policy has been violated.

42. This claim describes a method for protecting a network of computers from mobile code by having a gateway computer that intercepts (receives) all "Downloadable" programs destined for the client computers, checking to see if any of these programs have suspicious computer operations within them. The claim requires particular steps to perform these checks, where a list of (potentially) suspicious computer operations, derived from the Downloadable, is compared to a security policy to identify if any of these are problematic. If so, the program is blocked (prevented), otherwise it is passed along to the client.

Shaio et al (U.S. Patent No. 6,571,338) (filed Dec. 20, 1995)

43. Shaio discloses an "intelligent" firewall that analyzes Java applets "to reduce the probability of viruses" by eliminating Java applets that contain instructions which do not conform to a pre-defined policy (*see* Shaio 5:7-11).
44. In particular, each of the steps in claim 1 of the '194 patent are disclosed by Shaio. First, Shaio's firewall receives a Java applet at a gateway. Further, the parties do not appear to dispute that a Java applet is a Downloadable under all construction. Second, Shaio's firewall compares a list of operations from the Java applet against a security policy. As an example of the comparison step, Shaio's Java bytecode verifier will look at the list of method calls performed by every class. This list will be verified to ensure that each method call targets a valid method which the callee is allowed to invoke. The security policy, as such, is the set of "public" and "private" labels place on method definitions throughout the Java system. Finally, Shaio's firewall prevents execution of the Java applet if the applet fails the security policy.

45. Consequently, Shaio anticipates Claim 1 of the '194 patent. Furthermore, as is detailed in Exhibit C, Shaio anticipates or renders obvious the remaining claims of the '194 patent.

The Firewall Toolkit (FWTK) 2.0 Beta release (revised March 1996 - Sept. 1996)

46. The Firewall Toolkit (FWTK) was one of the first application-layer firewalls and was introduced in the early 1990s. The FWTK was an open source firewall, so the source code was freely available and publicly distributed. In the Beta version of the FWTK, the architects of the firewall added functionality that allowed the firewall to inspect incoming information downloaded from the web in order to identify HTML tags that would instruct a user's web browser to automatically execute a Java Applet, ActiveX control, or JavaScript script. This functionality was added to an already rich feature-set of security controls in the FWTK.
47. Based on my review of the source code, each of the steps of Claim 1 of the '194 Patent is present in the FWTK product, with the exception of the creation of a list of suspicious computer operations. First, the FWTK intercepts, for example, a request for a Java applet (i.e. a Downloadable) before it is transferred to the client. Second, the FWTK compares a list of prohibited HTML operations to the operations present in the HTML code. For example, the "<applet>" tag instructs the browser to automatically run a Java applet within the requested webpage. If a user wishes to set a security policy that prohibits a web page from automatically running a Java applet, the FWTK scans the HTML code for an "<applet>" tag and removes it prior to execution by the browser. Third, the removal of the suspicious operation, e.g. the "<applet>" tag, prevents the browser from executing such an operation.
48. As I described in the exhibits, while I do not find a specific example of a "list" of

suspicious computer operations in the FWTK program, I understand that Finjan has accused WebWasher's product of containing a "list" of suspicious computer operations. Since I was unable to find a "list" of suspicious operations in the WebWasher program, my opinion regarding the FWTK product may change based on some broader reading of the term "list" that Finjan may construct.

49. Consequently, the FWTK renders Claim 1 of the '194 patent obvious. Furthermore, as is detailed in Exhibit D, the FWTK anticipates or renders obvious the remaining claims of the '194 patent. Finally, the FWTK may anticipate the claims of the '194 Patent based on Finjan's definition of a "list" based on their assertion that a "list" exists in WebWasher's program.

Combinations of Known Static Analysis Techniques and Gateway Detection

50. In addition to the single references that are cited above, I have summarized some references that when combined with one other reference, teach each element of all of the asserted claims of the '194 Patent. Again, a detailed analysis exists at Exhibit E-J.
51. I have separated the prior art with respect to these combinations into two categories: (1) static analysis of code to identify suspicious operations and (2) "gateway" prevention of hostile code. Two patents describe "gateway" detection of hostile code, and three references describe analyzing code to identify and list suspicious operation that may be performed by executable code. The combinations that render the '194 Patent obvious include each one-to-one combination of references between the two categories.

Static Analysis of Code to Identify Suspicious Operations (Lo 1991, Lo 1994, and Chen)

Summary of Static Analysis

52. As described in the background section on the state of the art, tools to statically analyze potentially malicious code to determine how code might behave when executed were

available and/or described at least as early as 1991. These analyzers, referred to as heuristic analyzers, would parse binary code and identify particular operations that may be deemed suspicious such as "File Write," "File Read," etc. Based on a combination of the identified operations in the code, this kind of tool could assist a user to identify and block hostile code.

Lo 1991

53. Lo 1991 describes a static analysis tool named "Snitch" that searches for suspicious operations such as e.g., "open, write, close" (p. 163). It performs this analysis by first disassembling the code and then examining it for the suspicious operations. If it identifies, for example, a duplicate system call operation (which would be unlikely to occur in a normal program), it reports this.

Lo 1994

54. Lo 1994 discusses a "malicious code filter" developed as a general-purpose testbed for research into the static analysis of code to detect malicious behaviors, investigating the use of sophisticated "program slicing" techniques to better understand how one part of a program might influence what is happening in another part of the program. For example, rather than simply identifying that a file is about to be read, Lo could potentially identify the name of the file that might be read. Using these techniques, Lo's tools can identify what Lo calls "tell-tale signs" of malicious behavior. The examples of tell-tale signs included a list of suspicious operations as is illustrated below (p. 5):

3.1. Tell-tale Signs Identified by Program Slicing

These tell-tale signs apply to all kinds of programs and are used with the program slicer.

- **File Read.** This includes the slicing for the `open()` system calls. The list of files being read will show what kind of information the program may access (e.g., strange accesses to `/dev/*` should be detected).
- **File Write.** In addition to the `open()` system call, it includes the uses of `create()`, `link()`, and `unlink()` system calls because a file modification can be simulated by deleting and creating a file. The files written to should be checked against a list of important system files (e.g., `/var/run/*`, `/etc/passwd`, `/etc/aliases`, `/bin/*`, `/usr/bin/*` files).²
- **Process Creation.** A malicious program uses the `fork()` system call to create processes. A denial-of-service malicious program may put a `fork()` system call in a loop to create a large number of processes.³
- **Program Execution.** A malicious program may create another process to perform the malicious action, so we check which other programs are invoked and examine them. Typical sequences are a `fork()` system call followed by an `exec()` system call, and the `system()` and `popen()` library calls.
- **Network Accesses.** Malicious programs can use the network to send information back to the writer. We will slice for the network system calls, e.g., `socket()`, `connect()`, `send()`.
- **Change of Protection State.** We slice for the change of protection-states system calls, e.g., `chmod()` and `chown()`. It is rather unusual for normal programs to use these system calls and could indicate the presence of a trojan horse.
- **Change of Privilege.** We slice for the `setuid()` and `setgid()` system calls.
- **Time-Dependent Computation.** We find out how the time is used in the program. A forward slicing on the `gettimeofday()` system call shows all variables that contain time-dependent values. We will slice again for the

Lo indicated that “by examining these signs, we can identify most malicious code” (p. 5).

Chen

55. Chen describes a macro virus scanner that can recognize both known and previously unknown macro viruses. Unknown macro viruses are identified by comparing suspicious instructions in the macro against a list of pairs of instructions known to be used by malicious viruses as shown in Chen’s Fig. 9 (reproduced below). If found, the macro can then be “treated” to remove the suspicious operations.

900

903 SET #	904 INSTRUCTION ID#	905 INSTRUCTION IDENTIFIER (TEXT/HEX)
902 1	1	.Format = 1 73 CB 00 0C 6C 01 00
	2	Macro Copy 67 C2 80
902 2	1	.Format = 1 73 CB 00 0C 6C 01 00
	2	Organizer .Copy 64 6F 02 67 DE 00 73 87 02 12 73 7F
3	1	.Format = 1 73 CB 00 0C 6C 01 00
	2	macros. 6D 61 63 72 6F 73 76 08
4	1	FileSaveAs a\$,1 12 6C 01 00
	2	MacroCopy 64 67 C2 80 6A 0F 47
5	1	ylformat c: /u" 79 7C 66 6F 72 6D 61 74 20 63 6A
	2	Environ\$ ("COMSPEC") 80 05 6A 07 043 4F 4D
.	.	.
.	.	.
.	.	.
902 i	1	...
	2	...

	j	...

"Gateway" Prevention of Hostile Code (Hershey and Ji 1995)

Summary of "gateway" prevention

56. As discussed in the state of the prior art, as computer networks gained in popularity, vendors and researchers began creating anti-virus and other security products that would operate within the network. One way of performing this task was to place previous anti-virus techniques on a device (often called a firewall) that intercepted incoming data from external servers before that data would be passed on to the internal client.

Hershey Summary

57. Hershey (U.S. Patent No. 5,414,833, filed Oct. 27, 1993) discloses a network firewall that can monitor the data flowing between internal and external devices. Hershey's "security agent" can detect and respond to attacks that it detects on the network. Hershey was concerned with monitoring the network to detect viruses.

Ji 1995 Summary

58. Ji (U.S. Patent No. 5,623,600, filed Sept. 26, 1995) discloses a network firewall that performs anti-virus scanning on network traffic between the users inside a corporate network and the outside world. Ji 1995 discusses how the virus-checking program might be "a program that performs a version of signature scanning virus detection" (7:60-61) or "other virus detection methods" (7:64). Ji 1995 describes various devices that can perform its disclosed method of virus detection. One of those devices is expressly named by Ji 1995 as a "gateway" (see, e.g., 2:35, 2:64, etc.).

One of Ordinary Skill Would Be Motivated to Combine These Approaches

59. There are several express references that would motivate one of ordinary skill in the art to combine each of the following references.

Hershey and Lo 1991

60. It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques,

including static analysis tools, “which have been shown to be effective against certain types of malicious code.” (5:7-9)

61. Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent references this specific article (5:3-10). Consequently, one of ordinary skill in the art would have been specifically motivated to combine it with Hershey.

Hershey and Lo 1994

62. It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that “it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks.” (4:54-58) Hershey then describes existing techniques, including static analysis tools, “which have been shown to be effective against certain types of malicious code.” (5:7-9)

63. Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent specifically references Lo 1991, completed by the same author, related to static analysis of malicious code. Consequently, one of ordinary skill in the art would have been specifically motivated to discover other work by Lo and to combine it with Hershey.

Hershey and Chen

64. It would have been obvious for one of ordinary skill in the art in 1996 to combine the

techniques disclosed in Chen with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that “it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks.” (Hershey 4:54-58) Hershey then describes existing techniques, including static analysis tools, “which have been shown to be effective against certain types of malicious code.” (Hershey 5:7-9)

65. Chen was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Chen patent specifically references Hershey. Consequently, one of ordinary skill in the art would have been specifically motivated to combine Chen and Hershey.

Ji 1995 and Lo 1991

66. It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji 1995. In particular, Ji 1995 itself discloses that “those skilled in the art will realize that various other virus detection methods may also be used.” (Ji 1995 7:63-65)
67. Lo 1991 was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji 1995.

Ji 1995 and Lo 1994

68. As discussed above, Ji 1995 itself discloses that “those skilled in the art will realize that various other virus detection methods may also be used” (7:63-65). Lo 1994 is another

example of a virus detection system that was known at the time. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji 1995.

Ji 1995 and Chen

69. Again, Ji 1995 discloses that “those skilled in the art will realize that various other virus detection methods may also be used” (7:63-65). Chen is yet another example of a virus detection system that was known at the time. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji 1995.

Combining One of the Gateway References With One of the Static Analysis References Renders Obvious the Asserted Claims of the ‘194 Patent

70. Recall that the ‘194 patent claim 1 recites:

A computer-based method, comprising the steps of:
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and
preventing execution of the Downloadable by the client if the security policy has been violated.

71. The gateway references, Ji 1995 and Hershey, both disclose *receiving* Downloadable information at a gateway.
72. The static analysis references, Lo 1991, Lo 1994, and Chen each disclose *comparing* a list of suspicious computer operations to a security policy.
73. All of the references identify protection techniques intended to *prevent* the execution of potentially malicious code.
74. Consequently, claim 1 of the ‘194 patent is invalid because it is obvious in light of the prior art. Likewise, the asserted dependent claims are also invalid as being either anticipated or obvious in light of the prior art, as is detailed in Exhibits C-J.

Summary of the '780 Patent

75. The '780 patent is a continuation of the '194 patent, sharing the same specification. The only differences occur in the claims. The claims of the '780 patent generally related to systems and methods for identifying particular pieces of code. The patent describes using a "Downloadable ID" as the mechanism to identify the code. In order to generate the Downloadable ID, "a hashing function" must be performed "on the Downloadable and the fetched software components to generate a Downloadable ID." (Claim 1)

State of the Prior Art

76. Hashing functions were well known at the time of the invention. Even one of Finjan's inventors on the '822 patent recognizes that hashing functions have been well-recognized since the 1980s and can be used to uniquely identify data. (Edery Depo., Oct. 28, 2007 at 115:11-116:6, rough depo.)
77. Broadly, hashing functions take arbitrarily long strings as inputs and produce a fixed-size output. Hashing functions, as such, produce a "summary" or "digest" of the original string. If two digests are the same, this does not necessarily imply that the original long strings are the same. However, if two digests are different, then the original long strings are necessarily different. "Secure" hashing functions have other important properties that are relied upon in various cryptographic contexts which are beyond the scope of this analysis.
78. Hashing has been applied to code in a commercial setting at least as early as 1996. Microsoft's "Authenticode" (used to sign ActiveX objects as part of Internet Explorer 3.0, released in 1996) as well as Sun and Netscape's "signed JAR files" (used to sign Java classes, also disclosed in U.S. patent 6,263,442, filed May 30, 1996). Broadly speaking,

these technologies compute “digital signatures” of code that can be later validated to determine that a given piece of code was endorsed by a particular signer and has not changed since that endorsement was made. It was well-known to one of skill in the art at the time that digital signatures use hashing functions as one of the steps in computing a digital signature.

Comparison of the Prior Art to the Asserted Claims of the ‘780 Patent

79. The ‘780 patent includes four independent claims. My analysis indicates that the four independent claims are not meaningfully different for the purposes of this summary analysis. For the convenience of the court, I have summarized the limitations that exist in all four independent claims and have described why those limitations are obvious. A detailed element-by-element analysis can be found in Exhibit K.

80. Each of the independent claims contains, in one fashion or another, the following limitations (here, quoting from claim 1):

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references;
and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

81. With respect to the first two limitations of obtaining and fetching, any Java applet or ActiveX control that contains references to other components will satisfy these two limitations. For example, a Java applet class containing a reference to another class file is obtained by the client. The referenced class file will be subsequently fetched by the client.

82. In the prosecution history, Finjan took the position that its claims were limited to

performing a hashing function on a Downloadable together with its fetched software components. (Amendment and Response to Office Action, sent on July 31, 2003, at 7)

83. I understand that Finjan has now taken the position in its Markman briefing that no such limitation is present. (Finjan's Answering Brief, at 12)
84. If Finjan takes the position that merely hashing on Java applets or ActiveX controls is novel, Authenticode and Signed Java anticipate that reading.
85. If Finjan takes the position that hashing a Java applet along with other referenced code to generate a single ID is novel, then it is my opinion that Authenticode and Signed Java render this claim obvious.
86. Consequently, the prior art anticipates and/or renders obvious all the claims of the '780 patent.

Summary of the '822 Patent

87. The '822 patent is related generally to methods and systems for instrumenting or otherwise wrapping code prior to its execution. The instrumentation has the effect of monitoring the execution of the wrapped code as it runs, allowing dangerous operations to be detected and/or prevented. For example, in an attempt to access the file system, the file name would be known at runtime and could then be subject to more detailed security policies that could not have been performed by statically analyzing the code in advance.

Ji, U.S. Patent No. 5,983,348 (filed Sept. 10, 1997)

88. Ji 1997, like the '822 patent, discloses methods and systems for instrumenting code prior to its execution. Ji discloses a firewall or proxy server that intercepts code on its way from the server to the client, allowing it to wrap possibly dangerous calls with pre-filters and post-filters, which allows those dangerous calls to be blocked. An example from

column 6 of the Ji 1997 specification of these pre and post-filters is provided below.

Examples of pre- and post-monitor functions are:

(1) to disallow any directory listing access:

```

pre-filter(function_name, parameters)
{
    if (function_name == "java.io.File.list")
        throw new SecurityException();
    }
    post-filter(result)
    {
    }
}

```

(2) To protect files under c:\temp from directory listing access:

```

pre-filter(function_name, parameters)
{
    if
    (function_name == "java.io.File.list")
    {
        extract the name of the file to be read from
        parameters;
        if the directory to be listed is under c:\temp)
            throw new SecurityException();
    }
    }
    post-filter(result)
    {
    }
}

```

Comparison of Prior Art to the Asserted Claims of the '822 Patent

89. The '822 patent has six independent claims. Claim 1 is representative of the limitations of all of the independent claims. However, some of the independent claims include minor additions or alterations. The claim chart in Exhibit L provides a detailed analysis of each of the particular differences, if any, between all of the independent and dependent claims asserted.

90. Claim 1 of the '822 patent recites:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the determining comprises *performing* one or more analyses of the downloadable-information, the analyses *producing* detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and *evaluating* the detection-indicators to determine whether the downloadable-information includes executable code.

91. I understand that the parties still dispute various terms in the '822 patent. I address those disputes in the claim chart (Exhibit L). Suffice it to say, for this summary analysis, that for each disputed term, Ji 1997 discloses elements that would meet the narrowest construction currently before the court. In other words, regardless of which party's construction is used, my opinion remains the same, namely that Ji 1997 anticipates all of the claims in the '822 patent. If the court's claim construction changes any of these terms, I may supplement this report to the extent that such construction changes my analysis.

92. Ji 1997 discloses all of the elements. Ji 1997 discloses *receiving* Java applets and ActiveX controls. Ji 1997's applet scanner must necessarily be able to *determine* whether the data it receives is executable code and, thus, must inherently *produce* indicators of when it is dealing with code versus other types of data and must inherently *evaluate* these indicators to determine when, in fact, it is dealing with executable code. This functionality is inherently necessary because, otherwise, Ji 1997 would be unable to perform the instrumentation (e.g., pre-filter and post-filter operations) that it describes and for which I have shown the example pre and post-filters, above, from the Ji 1997 specification. Ji 1997 describes *causing* a "live agent (e.g., a security monitoring package)" to be communicated to the client. Ji 1997's live agent is implemented by the

pre and post-filters.

93. Consequently, Ji 1997 anticipates the following claims of the '822 patent: 1, 4, 6, 8, 9, 12, 13, 15, 16, 17, 18, 20, 21, 22, 24, 26, 27, 28, 29, 31, 32, 34, and 35. Furthermore, Ji 1997 anticipates and/or renders obvious claim 2 of the '822 patent.

94. Counsel also asked me to look in the specification for the '822 patent to see if I could find structure related to the following terms that would enable one of ordinary skill to understand how to construct the particular functional component. The terms were: "mobile code means," "content inspection engine," "packaging engine," "linking engine," "transfer engine," "inspection controller," "MPC generator," and "policy generator." I was unable to find structure associated with these terms. With respect to my invalidity analysis, I referred to elements in the prior art that performed the functions described instead of looking for any particular structure. If the court determines that these terms have a particular structure, I may supplement this report to the extent that it changes any of my opinions.

XI. SUMMARY OF OPINIONS RELATED TO INFRINGEMENT

95. For the convenience of the Court, I will provide here a summary of my opinions regarding the infringement of the asserted '361 Patent and '010 Patent claims. Complete details of these opinions and supporting documentation are included in Exhibits M-N to this report.

XII. LEGAL STANDARDS FOR EVALUATING INFRINGEMENT

96. Counsel has instructed me that I am to rely on the following legal standards.

97. Counsel has instructed me that direct infringement under 35 U.S.C. § 271(a) requires that

a person make, use, sell, or offer to sell the claimed invention in the United States within the term of the patent. Counsel has instructed me that to find direct infringement, I must find each and every element or step of a claim in the accused product or process.

98. Counsel has instructed me that inducement to infringe under 35 U.S.C. § 271(b) requires that the alleged infringer must take actions that induce another to directly infringe and must know or should know that those actions would induce actual infringements. That is, the alleged inducer must have an affirmative intent to cause direct infringement. Counsel has instructed me that for inducement to occur there must be underlying direct infringement by another person, such as a customer following the instructions for using the Finjan product.

XIII. INFRINGEMENT OF THE '361 PATENT

Overview of '361 Patent

99. The '361 patent generally describes various systems and methods that allow a firewall to set security policies based on the corporate directory. One of the particular embodiments described in the patent describes a firewall that uses the Lightweight Directory Access Protocol (LDAP protocol) to access a corporate directory, such as a Microsoft Active Directory server. The firewall can then utilize the information contained in Active Directory that describes the user, for example the user's name and departments to which he/she belongs. This information may be useful for the firewall, because the firewall can base its security policies on a particular user or group to which the user belongs. If the firewall is able to communicate with the corporate directory, then any updates to the central directory can be used by the firewall without having to manually update a separate directory on the firewall. In this way, the invention reduces the duplication of effort that

was previously required to maintain a list of users and groups on the firewall itself.

Overview of the Vital Security NG appliances

100. I have reviewed the source code of the Vital Security NG-appliances as well as analyzed how one such appliance operated. I have also read several technical manuals regarding the operation of the Vital Security NG appliances. A complete list of materials that I have reviewed can be found at Exhibit B.
101. Generally, the Vital Security NG appliances are devices that perform various firewall services for small to large businesses. These firewall services are directed at securing the information coming into and going out of a corporation's network.
102. Based on my review of the source code and deposition testimony, I understand that Finjan sells and markets several types of NG-appliances that all have the same software installed on them. (Ben-Itzhak Depo. Nov. 2, 2007 (Rough Transcript) at 80-82) Finjan's different NG appliances differ only based on their respective hardware specifications as opposed to the software functionality built into the devices. (Ben-Itzhak Depo. Nov. 2, 2007 (Rough Transcript) at 80-82) Likewise, the documentation describing the functionality of the various NG appliances, e.g. NG-1000, NG-5000, and NG-8000 is all in one manual.
103. One particular feature of the NG-appliances is that it allows an administrator to import information, such as Usernames and Groups, from an Active Directory server. It accesses the information in the Active Directory server using the LDAP protocol. It then allows an administrator to set specific security policies based on the information stored in the Active Directory server.

Summary of opinions regarding infringement of the '361 Patent

104. As set forth in the charts, I find that the Finjan Vital Security NG appliances literally infringe the following apparatus claims of the '361 Patent: 1, 2, 3, 4, 5, 7, and 15.
105. Further, Finjan's conduct literally infringes and induces infringement of the following method claims: 8, 9, 10, 11, 12, and 14.
106. Counsel has informed me that, with respect to method claims, the accused infringer must perform the method at least once in order to infringe. I understand that Finjan has likely performed the claimed methods by operating their own Vital Security NG appliance products. Furthermore, I base my infringement analysis in part on several technical user manuals in which the instructions direct the user or purchaser of the product to use the product in such a way as to perform all of the limitations included in the asserted method claims.
107. I have included a detailed element-by-element analysis of the asserted claims of Secure Computing's '361 patent, attached as Exhibit M. I find every limitation of the asserted claims of the '361 Patent in the Vital Security NG appliances as detailed in the exhibit.

XIV. INFRINGEMENT OF THE '010 PATENT

Overview of '010 Patent

108. The '010 Patent discloses methods and systems for controlling access to confidential internal documents within a network. In particular, the '010 Patent describes a system that allows administrators to protect access to confidential documents based on a particular user or group with which the user is associated.

Overview of Finjan Mirage (Vital Security for Enterprise Documents)

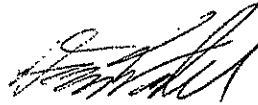
109. I was not given a copy of the source code for Finjan Mirage (also known as Vital Security

for Enterprise Documents and Vital Security for Documents) (hereafter "Enterprise Documents"). Likewise, I understand that Finjan produced very few documents related to Enterprise Documents. If I am given access to the source code or given further documents related to Enterprise Documents, I will supplement my report if the new information affects my opinion.

110. Broadly, the Enterprise Documents product is intended to impose access controls and other restrictions on documents. The product allows an administrator to specify policies for which users or groups of users are allowed to access a particular document. The product manuals explicitly describe network services, available to users both within and outside the organization deploying the product. Enterprise Documents is meant to sit in front of a normal Web server, adding new access controls onto these existing documents.

Summary of opinions regarding infringement of the '010 Patent

111. A complete element-by-element analysis of claim 37 of the '010 patent is included in Exhibit N to this report. I find every limitation of the asserted claim 37 in the Enterprise Documents as detailed in the exhibit.



21 Nov 2007

Dan Wallach

EXHIBIT A

Dan Seth Wallach

Home: 650-365-3973

Work: TBD

Fax: 650-859-2844

dwallach@cs.rice.edu

<http://www.cs.rice.edu/~dwallach/>

Department of Computer Science
Rice University
Duncan Hall 3121
6100 Main Street
Houston, TX 77005

Education **Princeton University** (Princeton, NJ), Department of Computer Science,
Ph.D. Computer Science, January 1999.
M.A. Computer Science, May 1995.
U.C. Berkeley (Berkeley, CA), College of Engineering,
B.S. Electrical Engineering/Computer Science, May 1993.

Publications (see also, [publications by area](#))

Journal Papers

Cristian Coarfa, Peter Druschel, Dan S. Wallach, [Performance Analysis of TLS Web Servers](#), *ACM Transactions on Computer Systems*, vol. 24, no. 1, February 2006.

Eyal de Lara, Yogesh Chopra, Nilesh Vaghela, Rajnish Kumar, Dan S. Wallach, Willy Zwaenepoel, [Iterative Adaptation for Mobile Clients Using Existing APIs](#), *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, October 2005 (also appeared in *IEEE Distributed Systems Online*, vol. 6, no. 9, September 2005).

Adam B. Stubblefield, Aviel D. Rubin, and Dan S. Wallach, [Managing the Performance Impact of Web Security](#), *Electronic Commerce Research Journal*, February, 2005.

Andrew M. Ladd, Kostas E. Bekris, Algis Rudys, Lydia E. Kavraki, and Dan S. Wallach, [Robotics-Based Location Sensing Using Wireless Ethernet](#), *Wireless Networks*, volume 11, number 1-2, January 2005, pp. 189-204.

Andrew M. Ladd, Kostas E. Bekris, Algis P. Rudys, Dan S. Wallach, and Lydia E. Kavraki, [On the Feasibility of Using Wireless Ethernet for Indoor Localization](#), *IEEE Transactions on Robotics and Automation*, volume 20, number 3, June 2004.

Y. Charlie Hu, Weimin Yu, Alan Cox, Dan S. Wallach, and Willy

Zwaenepoel, Runtime Support for Distributed Sharing in Safe Languages, *ACM Transactions on Computer Systems*, volume 21, number 1, pp. 1-35, February 2003.

Algis Rudys and Dan S. Wallach, Termination in Language-based Systems, *ACM Transactions on Information and System Security*, volume 5, number 2, May 2002.

Dan S. Wallach, Edward W. Felten, and Andrew W. Appel, The Security Architecture Formerly Known as Stack Inspection: A Security Mechanism for Language-based Systems, *ACM Transactions on Software Engineering and Methodology*, volume 9, number 4, October 2000.

Refereed Conference Papers

Atul Singh, Tsuen-Wan "Johnny" Ngan, Peter Druschel, and Dan S. Wallach, Eclipse Attacks on Overlay Networks: Threats and Defenses, *IEEE INFOCOM '06 (Barcelona, Spain)*, April 2006.

Animesh Nandi, Tsuen-Wan "Johnny" Ngan, Atul Singh, Peter Druschel, and Dan S. Wallach, Scrivener: Providing Incentives in Cooperative Content Distribution Systems, *ACM/IFIP/USENIX 6th International Middleware Conference (Middleware 2005) (Grenoble, France)*, November 2005.

Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavvaki, Practical Robust Localization over Large-Scale 802.11 Networks, *Tenth ACM International Conference on Mobile Computing and Networking (MOBICOM 2004) (Philadelphia, Pennsylvania)*, September 2004.

Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, Dan S. Wallach, Analysis of an Electronic Voting System, *2004 IEEE Symposium on Security and Privacy (Oakland, California)*, May 2004.

Scott Crosby and Dan S. Wallach, Denial of Service via Algorithmic Complexity Attacks, *12th Usenix Security Symposium (Washington, D.C.)*, August 2003.

David W. Price, Algis Rudys, and Dan S. Wallach, Garbage Collector Memory Accounting in Language-Based Systems, *2003 IEEE Symposium on Security and Privacy (Oakland, California)*, May 2003.

Eyal de Lara, Rajnish Kumar, Dan S. Wallach, and Willy Zwaenepoel, Collaboration and Multimedia Authoring on Mobile Devices, *First International Conference on Mobile Systems, Applications, and Services (MobiSys '03) (San Francisco, California)*, May 2003.

Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach, Security for Peer-to-Peer Routing Overlays, *Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*

(Boston, Massachusetts), December 2002.

Andrew M. Ladd, Kostas E. Bekris, Guillaume Marceau, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki, Using Wireless Ethernet for Localization, *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)* (Lausanne, Switzerland), October 2002.

Andrew M. Ladd, Kostas E. Bekris, Guillaume Marceau, Algis Rudys, Lydia E. Kavraki and Dan S. Wallach, Robotics-Based Location Sensing using Wireless Ethernet, *Eighth ACM International Conference on Mobile Computing and Networking (MOBICOM 2002)* (Atlanta, Georgia), September 2002.

Algis Rudys and Dan S. Wallach, Transactional Rollback for Language-Based Systems, *The 2002 International Conference on Dependable Systems and Networks (DSN-2002)* (Washington, D.C.), June 2002.

Cristian Coarfa, Peter Druschel, and Dan S. Wallach, Performance Analysis of TLS Web Servers, *Network and Distributed Systems Security Symposium* (San Diego, California), February 2002.

Eyal de Lara, Dan S. Wallach, and Willy Zwaenepoel, HATS: Hierarchical Adaptive Transmission Scheduling, *Multimedia Computing and Networking 2002 (MMCN02)* (San Jose, California), January 2002.

Jason Flinn, Eyal de Lara, M. Satyanarayanan, Dan S. Wallach, and Willy Zwaenepoel, Reducing the Energy Usage of Office Applications, *IFIP/ACM International Conference on Distributed Systems Platforms - Middleware 2001* (Heidelberg, Germany), November 2001.

Scott A. Craver, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan S. Wallach, Drew Dean, and Edward W. Felten, Reading Between the Lines: Lessons from the SDMI Challenge, *10th Usenix Security Symposium* (Washington, D.C.), August 2001.

Eyal de Lara, Dan S. Wallach and Willy Zwaenepoel, Puppeteer: Component-based Adaptation for Mobile Computing, *3rd Usenix Symposium on Internet Technologies and Systems (USITS '01)* (San Francisco, California), March 2001.

Algis Rudys, John Clements, and Dan S. Wallach, Termination in Language-based Systems, *Network and Distributed Systems Security Symposium* (San Diego, California), February 2001.

Eyal de Lara, Dan S. Wallach and Willy Zwaenepoel, Opportunities for Bandwidth Adaptation in Microsoft Office Documents, *4th Usenix Windows Systems Symposium* (Seattle, Washington), August 2000.

Dan S. Wallach and Edward W. Felten, Understanding Java Stack Inspection, *1998 IEEE Symposium on Security and Privacy* (Oakland,

California), May 1998, pp. 52-63.

Dan S. Wallach, Dirk Balfanz, Drew Dean, and Edward W. Felten, Extensible Security Architectures for Java, *16th Symposium on Operating Systems Principles* (Saint-Malo, France), October 1997, pp. 116-128.
—*outstanding paper award*

Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach, Web Spoofing: An Internet Con Game, *20th National Information Systems Security Conference* (Baltimore, Maryland), October 1996.

Drew Dean, Edward W. Felten, and Dan S. Wallach, Java Security: From HotJava to Netscape and Beyond, *1996 IEEE Symposium on Security and Privacy* (Oakland, California), May 1996, pp. 190-200.

Dan S. Wallach, Sharma Kunapalli and Michael F. Cohen, Accelerated MPEG Compression of Dynamic Polygonal Scenes, *Computer Graphics, SIGGRAPH 1994* (Orlando, Florida), August 1994, pp. 193-196.

Articles, Book Chapters, Etc.

Dan S. Wallach, Texas must confront voting systems' flaws, *Austin American-Statesman*, September 2004.

Jonathan Bannet, David W. Price, Algis Rudys, Justin Singer, Dan S. Wallach, Hack-a-Vote: Demonstrating Security Issues with Electronic Voting Systems, *IEEE Security & Privacy Magazine*, volume 2, number 1, January/February 2004, pp. 32-37. Also reprinted by *ComputerUser*, March 2004.

Dan S. Wallach, Copy Protection Technology is Doomed, *IEEE Computer*, volume 34, number 10, October 2001, pp. 48-49.

Drew Dean, Edward W. Felten, Dan S. Wallach, and Dirk Balfanz, Java Security: Web Browsers and Beyond, *Internet Beseiged: Countering Cyberspace Scofflaws*, D. E. Denning and P. J. Denning, Eds. ACM Press, New York, Oct. 1997, pp. 241-269.

PhD Dissertation

Dan S. Wallach, A New Approach to Mobile Code Security, PhD Dissertation, Princeton University, January 1999.
Advised by Edward W. Felten

Workshop Papers

Daniel Sandler and Dan S. Wallach, Casting Votes in the Auditorium, *2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)* (Boston, Massachusetts), August 2007.

Seth James Nielson, Scott A. Crosby, and Dan S. Wallach, A Taxonomy of Rational Attacks, *Fourth International Workshop on Peer-to-Peer*

Systems (IPTPS '05) (Ithaca, New York), February 2005.

Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach, AP3: Cooperative, Decentralized Anonymous Communication, *11th ACM SIGOPS European Workshop* (Leuven, Belgium), September 2004.

Tsuen-Wan "Johnny" Ngan, Animesh Nandi, Atul Singh, Dan S. Wallach, and Peter Druschel, Designing Incentives-Compatible Peer-to-Peer Systems, *2nd Bertinoro Workshop on Future Directions in Distributed Computing (FuDiCo 2004)* (Bertinoro, Italy), June 2004.

Tsuen-Wan "Johnny" Ngan, Dan S. Wallach, and Peter Druschel, Incentives-Compatible Peer-to-Peer Multicast, *2nd Workshop on Economics of Peer-to-Peer Systems* (Cambridge, Massachusetts), June 2004.

Ping Tao, Algis Rudys, Andrew Ladd, and Dan S. Wallach, Wireless LAN Location Sensing for Security Applications, *ACM Workshop on Wireless Security (WiSe 2003)* (San Diego, California), September 2003.

Andrew Fuqua, Tsuen-Wan "Johnny" Ngan, and Dan S. Wallach, Economic Behavior of Peer-to-Peer Storage Networks, *Workshop on Economics of Peer-to-Peer Systems* (Berkeley, California), June 2003.

Alan Mislove, Charles Reis, Ansley Post, Paul Willmann, Peter Druschel, Dan S. Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, Luciana Arantes-Bezerra, POST: A Secure, Resilient, Cooperative Messaging System, *9th Workshop on Hot Topics in Operating Systems (HotOS IX)* (Lihue, Hawaii), May 2003.

Nathanael Paul, David Evans, Aviel D. Rubin, and Dan S. Wallach, Authentication for Remote Voting, *Workshop on Human-Computer Interaction and Security Systems* (Fort Lauderdale, Florida), April 2003.

Tsuen-Wan "Johnny" Ngan, Dan S. Wallach, and Peter Druschel, Enforcing Fair Sharing of Peer-to-Peer Resources, *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)* (Berkeley, California), February 2003.

Algis Rudys and Dan S. Wallach, Enforcing Java Run-Time Properties Using Bytecode Rewriting, *International Symposium on Software Security* (Tokyo, Japan), November 2002.

Dan S. Wallach, A Survey of Peer-to-Peer Security Issues, *International Symposium on Software Security* (Tokyo, Japan), November 2002.

Yuri Dotsenko, Eyal de Lara, Dan S. Wallach, and Willy Zwaenepoel, Extensible Adaptation via Constraint Solving, *4th IEEE Workshop on Mobile Computing Systems & Applications* (Callicoon, New York), June

2002.

Eyal de Lara, Rajnish Kumar, Dan S. Wallach, and Willy Zwaenepoel, Collaboration and Document Editing on Bandwidth-Limited Devices, *Proceedings of the Workshop on Application Models and Programming Tools for Ubiquitous Computing (UbiTools '01)* (Atlanta, Georgia), September 2001.

Eyal de Lara, Dan S. Wallach, and Willy Zwaenepoel, Position Summary: Architectures for Adaptation Systems, *Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)* (Schloss Elmau, Germany), May 2001.

Y. Charlie Hu, Weimin Yu, Alan L. Cox, Dan S. Wallach, and Willy Zwaenepoel, Runtime Support for Distributed Sharing in Typed Languages, *Proceedings of LCR2000: the Fifth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers* (Rochester, New York), May 2000.

Dan S. Wallach, Jim A. Roskind, and Edward W. Felten, Flexible, Extensible Java Security Using Digital Signatures, *Network Threats* (New Brunswick, New Jersey), December 1996, R. N. Wright and P. G. Neumann, Eds., vol. 38 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, pp. 59-74.

Tech Reports / Miscellaneous

Srinivas Inguva, Eric Rescorla, Hovav Shacham, and Dan S. Wallach, Source Code Review of the Hart InterCivic Voting System, California Secretary of State's "Top to Bottom" Review, July 2007.

David L. Dill and Dan S. Wallach, Stones Unturned: Gaps in the Investigation of Sarasota's Disputed Congressional Election, April 2007.

Dan S. Wallach, "Expert Report in *Conroy v. Dennis*" (portions redacted), September 2006.

Dan S. Wallach, "Security and Reliability of Webb County's ES&S Voting System and the March '06 Primary Election" (Expert Report in *Flores v. Lopez*), May 2006.

Seth Nielson, Seth J. Fogarty, and Dan S. Wallach, Attacks on Local Searching Tools, Technical Report TR-04-445, Department of Computer Science, Rice University, December 2004.

Dan S. Wallach, Testimony for the NIST/EAC Technical Guidelines Development Committee (Gaithersburg, Maryland), September 2004.

Dan S. Wallach, Testimony for the Texas Senate Committee on State Affairs (Austin, Texas), May 2004.

Dan S. Wallach, Testimony for the Texas House Elections Committee (Austin, Texas), March 2004.

Dan S. Wallach, Testimony for the Ohio Joint Committee on Ballot Security (Columbus, Ohio), March 2004.

Jonathan Bannet, David W. Price, Algis Rudys, Justin Singer, Dan S. Wallach, Hack-a-Vote: Demonstrating Security Issues with Electronic Voting Systems, Technical Report TR-03-427, Department of Computer Science, Rice University, November 2003.

Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, Dan S. Wallach, Analysis of an Electronic Voting System, Johns Hopkins Information Security Institute Technical Report TR-2003-19, July 2003.

David L. Dill, Rebecca Mercuri, Peter G. Neumann, and Dan S. Wallach, Frequently Asked Questions about DRE Voting Systems (web page, also submitted to Santa Clara County board of supervisors), February 2003.

David W. Price, Algis Rudys, and Dan S. Wallach, Garbage Collector Memory Accounting in Language-Based Systems, Technical Report TR-02-407, Department of Computer Science, Rice University, December 2002.

Adam B. Stubblefield and Dan S. Wallach, Dagster: Censorship-Resistant Publishing Without Replication, Technical Report TR01-380, Department of Computer Science, Rice University, July 2001.

Alex Grosul and Dan S. Wallach, A Related-Key Cryptanalysis of RC4, Technical Report TR-00-358, Department of Computer Science, Rice University, June 2000.

Adam B. Stubblefield and Dan S. Wallach, A Security Analysis of My.MP3.com and the Beam-it Protocol, Technical Report TR-00-353, Department of Computer Science, Rice University, February 2000.

Eyal de Lara, Dan S. Wallach, and Willy Zwaenepoel, A Characterization of Compound Documents on the Web, Technical Report TR-99-351, Department of Computer Science, Rice University, November 1999.

Invited Panels

Steve Ansolabehere, *et al.*, Workshop on Developing a Research Agenda for Electronic Voting Technologies, American Association for the Advancement of Science (AAAS), September 2004.

Darleen Fisher, *et al.*, NSF Workshop on Security and Privacy (Berkeley, California), February 2002. *Publication pending.*

Gary McGraw, *et al.*, Attacking Malicious Code: A Report from the Infosec Research Council (San Antonio, Texas), April, 2000. Report

published in *IEEE Software* 17(5), pp. 33-40.

Teaching

Courses at Rice:

Comp527: Computer Systems Security (Spring 1999, Fall and Spring 2000, Fall 2001-2006)

Comp435: Election Systems, Technology, and Administration (Fall 2006)

Comp314: Applied Algorithms and Data Structures (Fall 1999, Spring 2001, 2002, 2004-2006)

Comp620: Seminar in Secure Systems (Fall 1998)

Short courses and tutorials:

Dan S. Wallach, *Software Engineering for Security* (a one-week intensive short course), presented at Secure Application Development (Leuven, Belgium), February 2007

Dan S. Wallach, *Language-Based Security* (a one-week intensive short course), presented at *The Summer School on Foundations of Internet Security* (Duszniki Zdrój, Poland), June 2002.

Dan S. Wallach and Drew Dean, *Java and Security* (a one-week intensive short course), Katholieke Universiteit Leuven (Leuven, Belgium), March 1997.

Teaching assistant positions at Princeton:

Introduction to Computer Systems (Spring 1996)

Computer Graphics (Fall 1993, Fall 1994, and Fall 1995)

Advanced Programming Techniques (Spring 1994)

Professional Service

Research management:

Associate Director, ACCURATE (NSF-funded research center), 2005-2010

Program committees:

ACM Conference on Computer and Communications Security (CCS) 2004 and 2005

ACM Conference on Electronic Commerce 2007

ACM Role-Based Access Control Workshop 1999 and 2000

Applied Cryptography and Network Security (ACNS) 2005

HotOS Workshop 2003

HotSec Workshop 2006

IEEE International Conference on Distributed Computing Systems (ICDCS) 2007

IEEE Security and Privacy 1999, 2004, 2005, 2007, and 2008

IEEE Workshop on Mobile Computing Systems and Applications (WMCSA) 2002 and 2004
3rd International Conference on Electronic Voting 2008
International Peer-to-Peer Symposium (IPTPS) 2004 and 2006
Network and Distributed Systems Security Symposium (NDSS) 2002-2004 and 2006
NSF grant panels 2002, 2004, 2005, 2006
South Central Information Security Symposium 2003-2006
Usenix/ACCURATE Electronic Voting Workshop 2007
Usenix Annual Conference 2001
Usenix Security Symposium 1999-2003 and 2005
Usenix Symposium on Internet Technologies and Systems (USITS) 2003
Workshop on Economics in Peer-to-Peer Systems 2004
WWW Conference 1999, 2000, 2003, 2004, 2006, and 2007

Program committee chair:

Usenix Security Symposium 2001
Usenix/ACCURATE Electronic Voting Workshop 2006
WWW Conference, Co-Chair of Security, Privacy, Reliability, and Ethics Track 2007-2008
Web 2.0 Security & Privacy Workshop (W2SP), Co-Chair 2007

Invited talks coordinator:

Usenix Security Symposium 2002

Panel moderator/organizer (electronic voting security):

Usenix Security Symposium 2003
IEEE Symposium on Security and Privacy 2004

Workshop organizer:

South Central Information Security Symposium 2003-2006

Editorial and advisory board memberships:

Election Science Institute (VoteWatch)
IEEE Internet Computing (2004-2006)
International Journal of Information Security
International Journal of Information and Computer Security
International Journal for Infonomics
National Committee for Voting Integrity
Verified Voting Foundation / VerifiedVoting.org

University committees:

Advisor for MCS Students (2000-2001)
CS Graduate Admissions (1998-2005)
CS Curriculum Committee (2005-present)

CS Facilities (occasional involvement)
KTRU (Rice Radio) Friendly Committee (2005-present)
University IT Security Committee (2002-present)

Other university service:

Divisional advisor and faculty associate, Martel College (2001-present)
Rice Social Dance Society: faculty sponsor, instructor, workshop organizer, etc. (2001-present)

Grants

Aviel D. Rubin, Dan S. Wallach, Michael Byrne, Douglas W. Jones, David Dill, Dan Boneh, David A. Wagner, Dierdre Mulligan, Drew Dean, and Peter G. Neumann, CT-CS: A Center for Correct, Usable, Reliable, Auditable, and Transparent Elections (ACCURATE), NSF CNS-0524211 (October 2005).

Dan S. Wallach and Peter Druschel, CSR/PDOS: Security and Incentives for Overlay Network Infrastructure, NSF CNS-0509297 (August 2005).

Dan S. Wallach and Mike Dahlin, Resource Management for Safe Deployment of Edge Services, Texas Advanced Technology Program #003604-0053-2001 (October 2001).

Dan S. Wallach, Security and Resource Management in Type-Safe Language Environments, NSF CAREER CCR-9985332 (March 2000).

Behnaam Aazhang, Richard G. Baraniuk, Joseph R. Cavallaro, Edward W. Knightly, and Dan S. Wallach, Seamless Multitier Wireless Networks for Multimedia Applications, NSF Special Projects ANI-9979465 (April 1999).

Industrial gifts and support:

Microsoft gift (November 2002)
Schlumberger gift (February 2002)
IBM University Partnership Program (June 2000)
Microsoft gift (July 2000)

Related support:

Usenix Student Scholarship for Adam Stubblefield (May 2001)

Invited Talks and Panels

Michael E. Clark, Joseph E. Savage, Peter Toren, and Dan S. Wallach, *Trade Secret and Confidential Information*, Panel at the ABA National Institute on Computing and the Law (San Francisco, California), June 2007.

Dan S. Wallach, *Testimony Before the Senate Committee on Rules and Administration, Hearing on Electronic Election Reform* (Washington, D.C.), February 2007.

Dan S. Wallach, *Electronic Voting: Risks and Research*, Institute for Security Technology Studies Distinguished Speaker Series, Dartmouth College (Hannover, New Hampshire), October 2006.

Dan S. Wallach, *Electronic Voting: Risks and Research*, Max Planck Institute for Software Systems (Saarbrücken, Germany), October 2006.

Dan S. Wallach, *Electronic Voting: Risks and Research*, , Chaire Internationale en Sécurité Informatique, Institut Eurécom (Sophia Antipolis, France), October 2006.

Dan S. Wallach, *Electronic Voting: Risks and Research*, University of Texas at Austin (Austin, TX), September 2006.

Dan S. Wallach, *The Risks of Electronic Voting*, Election Protection Summit (Washington, D.C.), June 2006.

Dan S. Wallach, *Computer Security Education at Rice*, Workshop on Information Assurance Education (Houston, Texas), May 2006.

Dan S. Wallach, *The Risks of Electronic Voting*, Georgia Institute of Technology (Atlanta, Georgia), March 2006.

Dan S. Wallach, *Testimony for the California Senate Elections, Reapportionment & Constitutional Amendments Committee* (Menlo Park, California), February 2006.

Elizabeth Hanshaw Winn and Dan S Wallach, *Panel: Electronic Voting Technology*, First Annual Legislative and Public Policy Conference, TSU Thurgood Marshall School of Law (Houston, Texas), October 2005.

Paul Craft, Douglas Jones, John Kelsey, Ronald Rivest, Michael Shamos, Dan Tokaji, Dan S. Wallach, *Panel: Threat Discussion on Trojan Horses, Backdoors, and Other Voting System Software-Related Problems*, NIST Workshop on Threats to Voting Systems (Gaithersburg, Maryland), October 2005.

Dan S. Wallach, *The Risks of Electronic Voting*, Virginia Joint Committee Studying Voting Equipment (Richmond, Virginia), August 2005.

Dan S. Wallach, *The Risks of Electronic Voting*, Tarrant County Democratic Party Meeting (Hurst, Texas), July 2005.

Dan S. Wallach, *Electronic Voting Machine / Registration Systems*, Testimony for the Carter-Baker Commission on Federal Election Reform (Houston, Texas), June 2005.

Dan S. Wallach, *The Risks of Electronic Voting*, NSF Workshop on Cyberinfrastructure and the Social Sciences (Arlington, Virginia), March 2005.

Dan S. Wallach, *The Risks of Electronic Voting*, CASSIS: Construction and Analysis of Safe, Secure, and Interoperable Smart Devices (Nice, France), March 2005.

Dan S. Wallach, *The Risks of Electronic Voting*, University of Massachusetts, Amherst, Five Colleges Information Assurance Lecture Series (Amherst,

Massachusetts), December 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, University of Iowa, Department of Computer Science (Iowa City, Iowa), December 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, CSI's 31st Annual Computer Security Conference (Washington, D.C.), November 2004.

Hans Klein, Eugene Spafford, Donald Moynihan, Dan S. Wallach, and Jim Reis, *Panel: E-Voting Policies and Perils*, Association for Public Policy Analysis and Management (APPAM) (Atlanta, Georgia), October 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Seventh Workshop on Languages, Compilers, and Run-time Support for Scalable Systems (Houston, Texas), October 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Symposium on the 2004 Presidential Election, John J. Marshall Law School (Chicago, Illinois), October 2004.

Chris Bell, Dan S. Wallach, and Tony J. Servello III, *Panel: Electronic Voting*, Science Café (Houston, Texas), October 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, The Integrity of the Election Process, U. of Toledo Law School (Toledo, Ohio), October 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Princeton University, Department of Computer Science (Princeton, New Jersey), October 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, DIMACS Workshop on Cryptography: Theory Meets Practice (Piscataway, New Jersey), October 2004.

Dan S. Wallach, Michael I. Shamos, Eugene Spafford, and Michael E. Lavelle, *Panel: Who Can Plug Into E-Voting Machines?*, E-lection 2004: Is E-Voting Ready for Prime Time?, John Marshall Law School (Chicago, Illinois), October 2004.

Dan S. Wallach, Testimony for the NIST/EAC Technical Guidelines Development Committee (Gaithersburg, Maryland), September 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, DiverseWorks: The Voting Machine (Houston, Texas), September 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Baker Institute Forum on Electronic Voting (Houston, Texas), September 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, League of Women Voters General Meeting (Houston, Texas), September 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Simposio acerca de Urnas Electrónicas para la Emisión del Voto Ciudadano (Mexico City, Mexico),

September 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Fermi National Accelerator Lab (Batavia, Illinois), August 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, TrueMajority "National Day of Action" (Austin, Texas), July 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, 10th Annual County and District Clerks' Association of Texas Conference (Lake Conroe, Texas), June 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Texas State Democratic Party Convention, Progressive Populist Caucus (Houston, Texas), June 2004.

Dan S. Wallach, *Hack-a-Vote: Demonstrating Security Issues with Electronic Voting Machines*, DIMACS Workshop on Electronic Voting - Theory and Practice (Piscataway, New Jersey), May 2004.

Dan S. Wallach, Testimony for the Texas Senate Committee on State Affairs (Austin, Texas), May 2004.

Josh Benaloh, Dana DeBeauvoir, and Dan S. Wallach. *Panel: Electronic Voting Security*, IEEE Symposium on Security and Privacy (Oakland, California), May 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Harris County Democrats (Houston, Texas), April 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, North Brazoria County Democrats (Pearland, Texas), April 2004.

Dana DeBeauvoir, Ann McGeehan, Dan S. Wallach, *Panel on the Security of Electronic Voting*, League of Women Voters (Austin, Texas), April 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Guest lecture in "Texas Political Parties and Elections" (Government 335N, University of Texas, Austin), March 2004.

Dan S. Wallach, Testimony for the Texas House Elections Committee (Austin, Texas), March 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Bell County Republican Convention (Belton, Texas), March 2004.

Dan S. Wallach, Testimony for the Ohio Joint Committee on Ballot Security (Columbus, Ohio), March 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Houston Peace Forum (First Unitarian Universalist Church, Houston, Texas), March 2004.

Ben Cohen and Dan S. Wallach, *TrueMajority Press Event* (Washington, D.C.)

February, 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, European Commission eDemocracy Seminar (Brussels, Belgium), February, 2004.

Dana DeBeauvoir, Dan S. Wallach, Ann McGeehan, Bill Stotesbery, Adina Levin, *Electronic Voting: Benefits & Risks*, First Unitarian Universalist Church of Austin (panel co-sponsored by Travis County Green Party and Austin Democracy Coalition) (Austin, Texas), January 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, Texas IMPACT / United Methodist Women (Austin, Texas), January 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, River Oaks Democratic Women (Houston, Texas), January 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, University of Michigan, Department of Computer Science (Ann Arbor, Michigan), January 2004.

Dan S. Wallach, *The Risks of Electronic Voting*, EFF-Austin Policy Roundtable (Austin, Texas), December 2003.

Dan S. Wallach, *O.S. Security Semantics for Language-based Systems*, Katholieke Universiteit Leuven (Leuven, Belgium), December 2003.

Dan S. Wallach, *O.S. Security Semantics for Language-based Systems*, Belgium Java User's Group: JavaPolis (Antwerp, Belgium), December 2003.

Dan S. Wallach, *The Risks of Electronic Voting*, Austin Pastoral Center (Austin, Texas), November 2003.

Dan S. Wallach, *Peer-to-Peer Security*, Cornell University, Department of Computer Science (Ithaca, New York), November 2003.

Dan S. Wallach, *The Risks of Electronic Voting*, Duke University, Department of Computer Science (Durham, North Carolina), October 2003.

Dan S. Wallach, *The Risks of Electronic Voting*, University of Arizona, Department of Computer Science (Tucson, Arizona), September 2003.

Dan S. Wallach, *Peer-to-Peer Security*, UW/MSR/CMU Software Security Summer Institute (Stevenson, Washington), June 2003.

Dan S. Wallach, *Peer-to-Peer Security*, Stanford University, Department of Computer Science (Stanford, California), May 2003.

Dan S. Wallach, *Adventures in Copy Protection Research*, The Hockaday School (Dallas, Texas), April 2003.

Dan S. Wallach, *Adventures in Copy Protection Research*, Formal Techniques for Networked and Distributed Systems (Houston, Texas), November 2002.

Dan S. Wallach, *Peer-to-Peer Security*, Oregon Graduate Institute (Portland,

Oregon), March 2002.

Dan S. Wallach, *Mobile Code Security Through Program Transformations*, Mathematical Foundations of Programming Semantics (New Orleans, Louisiana), March 2002.

Dan S. Wallach, *The Risks of E-Voting Machines*, Bay Area New Democrats (Houston, Texas), November 2001.

Dan S. Wallach, Testimony before the Houston City Council on the risks of electronic voting systems, July 2001.

Dan S. Wallach, *Adventures in Copy Protection Research*, Open Group Meeting (Austin, Texas), July 2001.

Dan S. Wallach, *Adventures in Copy Protection Research*, Houston Copyright Town Hall Meeting (Houston, Texas), April, 2001.

Dan S. Wallach, *Mobile Code Security Through Program Transformations*, U.C. Berkeley (Berkeley, California), March 2001.

Dan S. Wallach, *Mobile Code Security Through Program Transformations*, University of Texas (Austin, Texas), November 2000.

Dan S. Wallach, *Mobile Code Security Through Program Transformations*, International Workshop on Mobile Objects/Code and Security (Tokyo, Japan), October 2000.

Dan S. Wallach and John DeRose, *The Security of My.MP3.com and Other "Beaming" Technologies*, MP3 Summit (San Diego, California), June 2000.

Dan S. Wallach, *An Overview of Computer Security*, Law Practice Management Section of the Houston Bar Association (Houston, Texas), May 2000.

Drew Dean and Dan S. Wallach, *Java Security: HotJava to Netscape and Beyond*, Workshop on security and languages (Palo Alto, CA), October 1997.

Drew Dean and Dan S. Wallach, *Java Security: HotJava to Netscape and Beyond*, Bell Laboratories (Murray Hill, NJ), April 1996.

- Wallach has also spoken to visiting groups of high school students via a Rice outreach program organized by Jen Overton.

Advisees

Completed PhDs:

Eyal de Lara (PhD completed January '03, now a professor at the University of Toronto)

Algis Rudys (PhD completed March '07, now at Google)

Recent graduate researcher collaborators (* = Wallach is current, primary advisor):

Konstantinos Bekris
Cristian Coarfa
Scott Crosby*
Anwis Das (current at Google)
Yuri Dotsenko
Tadayoshi Kohno (professor at University of Washington)
Rajnish Kumar
Andrew Ladd
Alan Mislove
Animesh Nandi
Seth Nielson*
Tsuen Wan "Johnny" Ngan (currently at Symantec Research)
Ainsley Post
Dan Sandler*
Atul Singh
Bryan Smith*
Adam Stubblefield (Johns Hopkins)
Ping Tao

Recent undergraduate researchers:

Jonathan Bannet
Kyle Derr
Eliot Flannery
Andrew Fuqua
David Wray Price (JD from Stanford, now clerking for the Federal Circuit, Patent Appeals Court)
Adam Stubblefield (2002 CRA Outstanding Male Undergraduate Award)
Andy Thomas
Ted Torous

Consulting *Private Consulting:*

California Secretary of State (summer 2007, contracting through the University of California, working on the Voting System Review)

AT&T Research (fall 2001, collaborating with Avi Rubin on security research)

GalleryFurniture (August 2001, post-attack web site audit and reinstall)

Curl (December 2000, security architecture review)

Quaadros Technologies (October 2000, design review)

Cloakware (September 2000 and August 2001, design review)

Coral Technologies (December 1999, security audit)

MetaCreations (March 2000, security audit)

CenterPoint Ventures (ongoing, technical evaluations of startups)

Rho Ventures (ongoing, technical evaluations of startups)

Legal Consulting (Election-related):

Jennings v. Buchanan (November 2006, expert for plaintiffs)

Conroy et al. v. Dennis (Colorado Sec. of State) (September 2006, expert for plaintiffs)

Santana et al. v. Williams (Texas Sec. of State) and DeBeauvoir (Travis County Clerk) (July 2006, expert for plaintiffs)

Taylor et al. v. Cortés (Pennsylvania Sec. of Commonwealth) (April 2006, expert for plaintiffs)

Bruni v. Valdes and Benavides (April 2006, expert for Bruni)

Flores v. Lopez (April 2006, expert for Flores)

ACLU v. Connor (Texas Sec. of State) (February 2005, expert for the ACLU)

Legal Consulting (Other):

Autobytel v. Dealix (May 2005, expert for Dealix)

Soverain v. Amazon.com (April 2005, expert for Amazon.com)

Uniloc v. Microsoft (November 2004, expert witness for Microsoft)

Nash v. Microsoft (May 2004, expert witness for Microsoft)

Recruitsoft v. Hire.com (August 2003, expert witness for Hire.com)

DirecTV v. NDS (April 2003, expert witness for DirecTV)

RIAA v. MP3.com (February 2000, wrote declaration for MP3.com)

Employment History Rice University, Associate professor, Department of Computer Science, beginning October 1998. (Promoted from assistant professor in May 2005.)

1/07 - 12/07 Stanford University, CS Department, visiting faculty / SRI Computer Science Laboratory, visiting researcher

9/93 - 10/98 Princeton University, Graduate student, Department of Computer Science. Supported by grants from NSF, Sun Microsystems, Intel, Microsoft, and others.

6/97 - 8/97 Netscape Communications Corporation, Mountain View, California.
Integrated Java with SSL. Audited the CORBA and RMI implementations for security bugs. Wrote a CORBA demonstration (a chat server).

6/96 - 8/96 Netscape Communications Corporation, Mountain View, California.
Designed and implemented a privilege-based security mechanism and user interface to enable digitally-signed Java applets. Participated in design reviews

of several Netscape and JavaSoft technologies.

6/95 - 8/95 **Microsoft Corporation**, Redmond, Washington.

Wrote a converter from Softimage to a RenderMorphics-based system (**V-Chat**). Designed and implemented a polygonal model compression system for virtual reality applications.

6/94 - 8/94 **David Sarnoff Research Center**, Princeton, New Jersey.

Wrote a microcode-level simulator for parallel video processing engine. Wrote design documents for the client side of a future video-on-demand system.

6/93 - 8/93 **Berkeley Systems**, Berkeley, California.

Ported a screen-reading system (allowing blind people to use graphical user interfaces) from Microsoft Windows to X.

9/92 - 6/93 **U.C. Berkeley**, Research Assistant for Dr. Larry Rowe.

Implemented parts of a MPEG-1 video encoder. Wrote the audio support for a real-time distributed media-on-demand system.

EXHIBIT 1, PART 2

EXHIBIT B

EXHIBIT B: DOCUMENTS CONSIDERED BY DAN WALLACH

DOCUMENT DESCRIPTION	DOCUMENT DATE	BATES START NO.	BATES END NO.
U. S. Patent No. 6,092,194 and File History	07/18/00		
U. S. Patent No. 6,357,010 and File History	03/12/02		
U. S. Patent No. 6,804,780 and File History	10/12/04		
U. S. Patent No. 7,058,822 and File History	06/06/06		
U. S. Patent No. 7,171,681	01/30/07		
U. S. Patent No. 7,185,361 and File History	02/27/07		
U.S. Patent No. 6,167,520	12/26/00		
PCT 97/01626 File History			
U.S. Provisional Patent Application No. 60/205,591	05/17/00		
U.S. Patent No. 5,740,441	04/14/98	SC202300	SC202325
U.S. Patent No. 5,696,822	12/09/97	SC201508	SC201508
U.S. Patent No. 5,623,600	04/22/97	SC201471	SC201491
U.S. Patent No. 5,414,833	05/09/95	SC188684	SC188727
U.S. Patent No. 6,571,338	05/27/03	SC202564	SC202575
U.S. Patent No. 5,951,698	09/14/99	SC189335	SC189353
Finjan's Opening Claim Construction Brief	09/07/07		
Secure Computing's Opening Claim Construction Brief	09/07/07		
Finjan's Answering Claim Construction Brief	09/28/07		
Secure's Responsive Claim Construction Brief	09/28/07		
Markman Hearing transcript	10/24/07		
German Interview of Martin Stecher	08/28/07		
German Interview of Christoph Alme	08/29/07		
German Interview of Frank Berzau	08/31/07		
German Interview of Peter Borgotie	08/31/07		
Deposition of Steven Chew and Deposition Exhibits 1-2	09/07/07		
German Interview Exhibits 1-53	08/28/07		
Deposition of Michael Gallagher and Deposition Exhibits 1-21	08/03/07		
Yuval Ben-Itzhak Deposition	08/09/07		
Yuval Ben-Itzhak Deposition	08/10/07		
Yuaval Ben-Itzhak Deposition Exhibits 1001 - 1021	08/10/07		
Deposition of David Kroll and Deposition Exhibits 1-9	09/14/07		
Deposition of Paula Greve and Deposition Exhibits 1-3	09/12/07		
Rough Draft Deposition of Shlomo Touboul Vol. I	10/23/07		

EXHIBIT B: DOCUMENTS CONSIDERED BY DAN WALLACH

DOCUMENT DESCRIPTION	DOCUMENT DATE	BATES START NO.	BATES END NO.
Rough Draft Deposition of Shlomo Touboul Vol. II	11/01/07		
Rough Draft Deposition of Dan Frommer	10/30/07		
Rough Draft Deposition of Yuval Ben-Itzhak Vol. III	11/02/07		
Rough Draft Deposition of Yigal Edery	10/29/07		
Rough Draft Deposition of Nimrod Vered	10/28/07		
"Blocking Java Applets at the Firewall," Martin, et al.		SC200386	SC200396
"Towards a Testbed for Malicious Code Detection," Kerchen, et al.	1991	SC189227	SC189233
"MCF: A Malicious Code Filter," Lo, et al.	05/04/94	SC189272	SC189298
Anti-Virus Tools and Techniques for Computer Systems, Polk, et al. Chapter 4	1995	SC189299	SC189316
VIRUS: Detection and Elimination, Skardhamar Chapter 4)	1996	SC189317	SC189334
Computer Viruses And Anti-Virus Warfare - Second Revised Edition, Hruska Chapter 7	1992	SC189354	SC189364
VIRUS: Detection and Elimination, Skardhamar Chapter 4 (DUPE??)	1996	SC189365	SC189382
Anti-Virus Tools and Techniques for Computer Systems, Polk, et al. Chapter 4 (DUPE??)	1995	SC189383	SC189400
Newsbytes Developer Recants Hostile Java Applet Story	08/07/96	SC189401	SC189403
Dissertation by Raymond Waiman Lo	1992	SC189404	SC189594
Documents 1Box	2005	SC011256	SC011257
Vital Security for Enterprise Documents	2003	SC011258	SC011259
The Business Case for Deploying Mirage in Pharmaceutical Companies	12/00/01	SC011244	SC011255
"Combating Viruses Heuristically," Veldman	09/00/93	SC200423	SC200432
IDC - Finjan Software: Closing the Window of Vulnerability, Burke	10/00/03	FIN003455	FIN003459
Finjan Software Consolidated Management Reports - Unaudited Draft for Discussion Only		FIN009002	FIN009003
"Personal Security Assistance for Secure Internet Commerce (position paper)," Rasmusson, et al. Available as dvi, ps, html, at http://www.sics.se/~ara/papers/NSP96.html	09/16/96		
Firewall Toolkit "fwtk.tar.z" source code			
"Finjan : Vital Security for Documents," marketing material found at URL below, hereafter "Presence". http://www.presence-security.co.uk/index.cfm/page/products.details.cfm/id/196			

EXHIBIT B: DOCUMENTS CONSIDERED BY DAN WALLACH

DOCUMENT DESCRIPTION	DOCUMENT DATE	BATES START NO.	BATES END NO.
"Finjan Software Launches a New Product - Vital Security for SSL", press release dated July 19, 2004 at the URL below. Hereafter "PR". http://www.finjan.com/Pressrelease.aspx?PressLan=329&id=424&lan=3			
Firewall Toolkit "fwtk-2.0beta.tar.z" source code			
http://www.finjan.com/content.aspx?id=190			
Security Policies In Depth		FIN017335	FIN017453
User Guide		FIN008703	FIN008826
Finjan's Management Console Reference Guide 8.3.5		FIN016486	FIN016658
Finjan's Management Console Reference Guide 8.30		FIN016323	FIN016485
Finjan's Management Console Reference Guide 8.4.0		FIN016659	FIN016845
Finjan's Management Console Reference Guide 8.4.3		FIN016846	FIN017042
Finjan's Management Console Reference Guide 8.5.0		FIN017043	FIN017315
Finjan's Installation and Setup Guide 8.30		FIN015842	FIN015916
Finjan's Installation and Setup Guide 8.3.5		FIN015988	FIN016063
Finjan's Installation and Setup Guide 8.4.0		FIN016064	FIN016144
Finjan's Installation and Setup Guide 8.4.3		FIN016145	FIN016225
Finjan's Installation and Setup Guide 8.5.0		FIN016226	FIN016322

EXHIBIT C

Claim Language '194 Patent	Shaio (U.S. Patent No. 6,571,338)
1. A computer-based method, comprising the steps of:	Shaio discusses the use of computers.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	<p>Shaio discloses an "intelligent firewall that provides real-time security testing of network packets, which may include executable code, such as applets" (2:25-27)</p> <p>Under Secure Computing's proposed construction of the terms in the '194 patent, a Downloadable is a program or document containing an executable application program that can be downloaded from one computer to another computer. Shaio's "network packets, which may include executable code" meet this definition.</p> <p>Under Finjan's proposed construction, a Downloadable is a "program or document containing mobile code." Shaio's network packets, which may include executable code" also meet this definition.</p> <p>Shaio discusses network packets, which include the network address of the destination to which the packets are being transmitted, i.e., the client computer (see, e.g., Shaio 5:1-5). This satisfies "addressed to a client" under both Finjan's and Secure Computing's proposed constructions.</p> <p>Under Secure Computing's proposed construction of a "server that serves as a gateway to the client," there must be "a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client." Shaio discloses "SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch." (3:55-57) Shaio's Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p> <p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no construction, saying that the server is "simply 'an intermediate between the Internet and the user.'" With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio's firewall meets this</p>

	limitation.
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	<p>Shaio discloses “additional security may be provided by intelligent firewall... For example, a bytecode verifier may parse the executable code portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses.” (5:7-11)</p> <p>The ‘194 patent explains the concepts of security profile data and suspicious computer operations, saying “it will be appreciated that the code scanner ... may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.” (5:54-57) Shaio’s Java bytecode verifier takes a number of steps, detailed in U.S. patent 5,740,441, incorporated by reference. These steps include, for example, looking at the list of method call operations and field references in the bytecode and verifying that they are properly formed (i.e., they point to a valid class or method which is legal to reference – this is a security policy being enforced by the Java bytecode verifier).</p> <p>In this scenario, the list of bytecodes being verified by Shaio are an example of the Downloadable security profile data in the ‘194 patent. The rules being evaluated by the bytecode verifier are examples of the security policy in the ‘194 patent. As a result, Shaio’s bytecode verifier meets the limitations of this claim element.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	Shaio discloses “there is a need for an intelligent firewall that provides real-time security testing of network packets, which may include executable code such as applets, and determines the risk level, i.e., trust worthiness, of each packet before permitting a lower-risk subset of the network packets to execute on anyone of the secure computers” (2:25-31) which meets this limitation of the ‘194 patent.
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	Shaio discloses, “For example, a bytecode verifier may parse the executable portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses” (5:8-11) which meets this limitation of the ‘194 patent.
3. The method of claim 2, wherein the security policy includes an access	The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to

control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Shao’s Java bytecode verifier, as described in U.S. patent 5,740,441, incorporated by reference, discusses the validation of method calls and field references. Shao describes a list of criteria that are tested against Java bytecode files (18:49-54) including checking that class names, field and method references, and so forth have valid names. This structure in Shao is an example of an access control list to which the Downloadable security profile data is compared.</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft’s Authenticode and Sun’s Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	<p>A URL (uniform resource locator) is a string describing the location and name of content which may be downloaded. URLs typically take the form of http://somehost.com/some/name/, where “somehost.com” is a network address and “some/name/” is a file available from that server.</p> <p>Shao describes a firewall which makes determinations based on the “source address of an executable packet” (see, e.g., 2:64-65). A URL is common way to express a source address. As such, Shao satisfies this element of the ‘194 patent.</p>
6. The method of claim 5, wherein the known URL is a trusted URL.	<p>Shao discloses, “a source/destination network address is considered uncertain if there is no match between the network address and a list of pre-approved secured network addresses” (4:35-38).</p> <p>Shao discloses “if the firewall determines within the first degree of certainty that the source address is associated with anyone of the secured nodes, then the firewall permits the executable packet to proceed to the secured computer.” (3:3-6)</p> <p>URLs can describe the addresses of both the source and the secured nodes. As such, Shao satisfies this</p>

	element of the '194 patent.
7. The method of claim 5, wherein the known URL is an untrusted URL.	Shaio discloses "Conversely, if firewall ... is uncertain or determines that the source address of the packet is outside ... then the packet is rejected." (5:1-5) Again, URLs can describe the addresses. As such, Shaio satisfies this element of the '194 patent.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	Shaio discloses "the present invention provides a method and apparatus for determining the trust worthiness of executable packets, e.g., internet applets, being transmitted within a computer network." Shaio satisfies this element of the '194 patent.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	As above, Shaio discusses "executable packets, e.g., internet applets." ActiveX is another example of an executable packet. Shaio satisfies this element of the '194 patent.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	As above, Shaio discusses "executable packets, e.g., internet applets." JavaScript scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	As above, Shaio discusses "executable packets, e.g., internet applets." Visual Basic scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Shaio's bytecode verifier is applied to all executable packets, regardless of origin or destination. Shaio satisfies this element of the '194 patent.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	<p>Shaio discloses, "a source/destination network address is considered uncertain if there is no match between the network address and a list of pre-approved secured network addresses" (4:35-38).</p> <p>The presence of this uncertainty, in Shaio, allows for different security policies to be applied. Shaio thusly satisfies this element of the '194 patent.</p>
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	Shaio discloses "network addresses may include a prefix field and a machine field, with the prefix field identifying clusters of computer systems couples to the respective network nodes, and the machine field identifying computer systems within each cluster." (4:39-43) This mechanism allows machines to be grouped together for the purpose of expressing a security policy. Thus, Shaio satisfies this element of the '194 patent.
24. The method of claim 1, further	This limitation describes the behavior of signature-

<p>comprising the step of comparing the Downloadable against a known Downloadable.</p>	<p>based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)</p>
<p>25. The method of claim 24, wherein the known Downloadable is hostile.</p>	<p>This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)</p>
<p>26. The method of claim 24, wherein the known Downloadable is non-hostile.</p>	<p>One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of</p>

	the program.” (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	<p>Hershey discloses a computer system that “provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity.” (5:23-27)</p> <p>Hershey and Shaio both disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from both systems together in a single firewall solution.</p>
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has “whitelist” and “blacklist” functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has “whitelist” and “blacklist” functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	With FWTK, in the event of a security policy violation, the user will see their requested web page replaced with an “error” web page, explaining the details of the policy violation. It would be obvious for any other firewall to adopt the same features.

EXHIBIT D

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	FWTK 2.0beta (September 1996)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>FWTK, the firewall toolkit, runs on computers.</p> <p>FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX).</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>In my analysis, I have been unable to locate a comparison of a list to a security policy having been disclosed in the FWTK source code. I understand that Finjan has accused WebWasher's product of containing a comparison of a list of suspicious computer operations to a security policy. I was unable in my analysis of the WebWasher product to identify a comparison of a list of suspicious computer operations to a security policy. After Finjan identifies a comparison of a list of suspicious computer operations to a security policy in WebWasher's product, my analysis of the FWTK source code may change based on Finjan's interpretation of what constitutes a comparison of a list of suspicious computer operations to a security policy.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>FWTK optionally removes the suspicious operations (e.g., the applet, script, and object tags) from HTML, thereby preventing the execution of a forbidden Downloadable.</p>
<p>2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.</p>	<p>FWTK decomposes HTML into its individual elements, which comprise the Downloadable security profile data.</p>
<p>3. The method of claim 2, wherein the security policy includes an access control list and further</p>	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test."</p>

comprising the step of comparing the Downloadable security profile data against the access control list.	('194 patent, 8:26-28)
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	FWTK supports a variety of different access control policies (enabling or disabling Java, JavaScript, and/or ActiveX), and will compare HTML elements against these policies. This structure in FWTK is an example of an access control list to which the Downloadable security profile data is compared. Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts (because they use the same <script> tag as is used by JavaScript)
12. The method of claim 1, wherein the security	FWTK's default policy is applied regardless of the client.

policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host).
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	The host names, specified in the 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code. Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art.
25. The method of claim 24, wherein the known Downloadable is hostile.	To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4) This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code. Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has

	support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art.
	To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art.
	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27)
	Hershey and FWTK both disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from both systems together in a single firewall solution.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site.
30. The method of claim 1, further comprising the	In the event of a security policy violation, the user will see their

step of informing a user upon detection of a security policy violation.	requested web page replaced with an "error" web page, explaining the details of the policy violation.
---	---

EXHIBIT E

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Hershey (U.S. Patent 5,414,833) and Lo et al. (1991)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Hershey and Lo disclose computer-based methods.</p> <p>Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing "malicious code detection."</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it . . .", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of</p>

	<p>these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent references this specific article (5:3-10). Consequently, one of ordinary skill in the art would have been specifically motivated to combine it with Hershey.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p>
<p>2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.</p>	<p>Lo discloses decomposing a compiled program into its assembly-language form ("The first module, the disassembler, takes an executable program as input and produces the equivalent Motorola 68020 assembly language representation as out", p. 163). This disassembled program comprises the Downloadable security profile data.</p>

<p>3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.</p>	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Lo discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163). This comprises a list that includes the criteria necessary for the Downloadable to fail or pass the test.</p>
<p>4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.</p>	<p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
<p>5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.</p>	<p>FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
<p>6. The method of claim 5, wherein the known URL is a trusted URL.</p>	<p>FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
<p>7. The method of claim 5, wherein the known URL is an untrusted URL.</p>	<p>FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to

	one of skill in the art and would be obviously applicable to any other firewall. To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a copending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (19:40-41)

EXHIBIT F

FINJAN SOFTWARE V. SECURE COMPUTING

Charts for 6,092,194

Claim Language: '194 Patent	Hershey (U.S. Patent 5,414,833) and Lo et al. (1994)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p> <p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Both Hershey and Lo disclose computer-based methods.</p> <p>Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).</p> <p>Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these</p>

	<p>against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent specifically references previous work, completed by the same author, related to static analysis of malicious code. Consequently, one of ordinary skill in the art would have been specifically motivated to discover other work by Lo and to combine it with Hershey.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the	<p>Lo discloses a variety of "tell-tale signs" which can be determined from all kinds of programs by using its "program slicer". These</p>

Downloadable security profile data.	tell-tale signs include File Read, File Write, Program Execution, Network Access, and a variety of other items (p. 5-6). The process of determining these tell-tale signs is an example of decomposing the Downloadable into the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)
	Lo discloses "Some program properties allow us to discern malicious programs from benign programs easily with very high accuracy without the need to give a specification of the program. We call those properties <i>tell-tale</i> signs. The idea is to program a <i>filter</i> to identify these tell-tale signs." (p. 3) Lo describes a list of these tell-tale signs (p. 5-6). This structure in Lo is an example of an access control list to which the Downloadable security profile data is compared.
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.
	Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with

	these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)

25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	Lo discloses comparisons between a program and "a 'clean' copy of the program" (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a copending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FW/TK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FW/TK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (19:40-41)

EXHIBIT G

FINJAN SOFTWARE V. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Hershey (U.S. Patent 5,414,833) and Chen (U.S. Patent 5,951,698)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Hershey and Chen disclose computer-based methods.</p> <p>Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of</p>

ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.

Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.

Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs," discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).

It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (Hershey 4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (Hershey 5:7-9)

Chen was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Chen patent specifically references Hershey. Consequently, one of ordinary skill in the art would have been specifically motivated to combine Chen and Hershey.

<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (Hershey 17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>Chen also discloses techniques to prevent execution: "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file." (Chen 17:21-25)</p>
<p>2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.</p>	<p>Chen discloses the decomposition of a macro: "once a set of instruction identifiers is obtained by the macro virus scanning module ... the decoded macro is scanned..." (15:14-16). The "set of instruction identifiers" comprises the Downloadable security profile data.</p>
<p>3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.</p>	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Chen discloses the comparison of the decoded macro (an example of Downloadable security profile data) to an access control list, an example of which is provided in Fig. 9. See also: "The comparison data in the virus information module preferably includes several sets of instruction identifiers. Various combinations of suspect instructions may be detected using the sets of instruction identifiers. Various different macro virus enablement and/or macro virus reproduction instructions may be identified using each set of</p>

	instruction identifiers. The instruction identifiers are not restricted to macro virus enablement and reproduction instructions. For example, an instruction which causes the computer hard disk to be reformatted without verification and an instruction which changes the system settings to allow such reformatting without user notice could be used as a suspect instruction combination." (14:52-64)
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the	FWTK can filter out ActiveX controls. FWTK was a widely

Downloadable includes an ActiveX™ control.	known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Chen apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
25. The method of claim 24, wherein the known Downloadable is hostile.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obviously applicable to any other firewall.
	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program

27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	and "a 'clean' copy of the program." (p. 4) Hershey discloses a copending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (Hershey 19:40-41) Chen discloses "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus." (Chen 17:21-23)

EXHIBIT H

FINJAN SOFTWARE V. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Lo et al. (1991)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Lo disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Lo discloses techniques for performing "malicious code detection."</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this</p>

comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

	<p>construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p>

	By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	Lo discloses decomposing a compiled program into its assembly-language form ("The first module, the disassembler, takes an executable program as input and produces the equivalent Motorola 68020 assembly language representation as out.", p. 163). This disassembled program comprises the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28) Lo discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163). This comprises a list that includes the criteria necessary for the Downloadable to fail or pass the test.
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other

	firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	It and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obviously applicable to any other firewall.
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4) Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27)
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	Hershey, Ji, and Lo disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each systems together in a single firewall solution. FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the	In the context of a FTP file transfer, when a virus is detected, Ji

step of informing a user upon detection of a security policy violation.	discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)
---	---

EXHIBIT I

FINJAN SOFTWARE V. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Lo et al. (1994)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Lo disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this</p>

	<p>construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to</p>

	a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	By deleting files with viruses rather than transferring them to the client, Ji prevents their execution. Lo discloses a variety of "tell-tale signs" which can be determined from all kinds of programs by using its "program slicer". These tell-tale signs include File Read, File Write, Program Execution, Network Access, and a variety of other items (p. 5-6). The process of determining these tell-tale signs is an example of decomposing the Downloadable into the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28) Lo discloses "Some program properties allow us to discern malicious programs from benign programs easily with very high accuracy without the need to give a specification of the program. We call those properties <i>tell-tale</i> signs. The idea is to program a <i>filter</i> to identify these tell-tale signs." (p. 3) Lo describes a list of these tell-tale signs (p. 5-6). This structure in Lo is an example of an access control list to which the Downloadable security profile data is compared.
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the	FWTK supports a "url-filter" feature, from a configuration file,

step of comparing the URL from which the Downloadable originated against a known URL.	which satisfies this limitation. FWTK also supports a “-dest” feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK’s “-dest” can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK’s “-dest” can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Ji and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the ‘hosts’ configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.

14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	Lo discloses comparisons between a program and "a 'clean' copy of the program" (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27)
	Hershey, Ji, and Lo disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each systems together in a single firewall solution.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	In the context of a FTP file transfer, when a virus is detected, Ji discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)

EXHIBIT J

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Chen (U.S. Patent 5,951,698)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Chen disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p>

	<p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Firjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves</p>

	<p>the configuration file to determine the handling of the temporary file.” (9:6-11)</p> <p>Ji also discloses “a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus.” (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: “Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	<p>Chen discloses the decomposition of a macro: “once a set of instruction identifiers is obtained by the macro virus scanning module ... the decoded macro is scanned...” (15:14-16). The “set of instruction identifiers” comprises the Downloadable security profile data.</p>
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Chen discloses the comparison of the decoded macro (an example of Downloadable security profile data) to an access control list, an example of which is provided in Fig. 9. See also: “The comparison data in the virus information module preferably includes several sets of instruction identifiers. Various combinations of suspect</p>

	instructions may be detected using the sets of instruction identifiers. Various different macro virus enablement and/or macro virus reproduction instructions may be identified using each set of instruction identifiers. The instruction identifiers are not restricted to macro virus enablement and reproduction instructions. For example, an instruction which causes the computer hard disk to be reformatted without verification and an instruction which changes the system settings to allow such reformatting without user notice could be used as a suspect instruction combination." (14:52-64)
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros. Ji and Chen apply their security policy regardless of the client to whom the Downloadable is addressed.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
25. The method of claim 24, wherein the known Downloadable is hostile.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obvious to apply to any firewall

	system.
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4) Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27)
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	Hershey, Ji, and Chen disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each system together in a single firewall solution. FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	In the context of a FTP file transfer, when a virus is detected, Ji discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (3:28-31) Chen discloses "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus." (Chen 17:21-23)

EXHIBIT 1, PART 3

EXHIBIT K

Claim Language '780 Patent	Microsoft's Authenticode (Microsoft Corp., Microsoft Authenticode Technology: Ensuring Accountability and Authenticity for Software Components on the Internet (Oct. 1996)) and Signed Java (U.S. Patent 6,263,442, filed May 30, 1996)
1. A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising:	Both Authenticode and Signed Java disclose digital signatures of code, which necessitates computing a hash function over that code, which is an example of the Downloadable ID.
obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;	Authenticode and Signed Java both include receiving executable code (e.g., ActiveX controls and Java applets). Both ActiveX and Java code include references from one software component to another. This is well known to one of ordinary skill in the art and recognized in the '780 patent itself.
fetching at least one software component identified by the one or more references; and	It is well known to one of ordinary skill in the art that when ActiveX and Java code include references to software components (e.g., other java class files or ActiveX controls) that those components will be fetched by the client.
performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.	Authenticode and Signed Java both disclosed a method of identifying a particular piece or combination of code. In order for Authenticode and Signed Java to generate an identifier to identify the particular code, Authenticode and Signed Java computed a digital signature, which included a performing a hashing function on the code. Because both Authenticode and Signed Java disclose hashing ActiveX controls and Java applets, respectively, it would be obvious to one of ordinary skill in the art that other arbitrary code, be it referenced or not, could also be included in a hashing function to identify the particular set of code hashed. If Finjan takes the position that merely hashing on Java applets or ActiveX controls is novel, Authenticode and Signed Java anticipate that reading. If Finjan takes the position that hashing a Java applet along with other referenced code to generate a single ID is novel, then it is my opinion that Authenticode and Signed Java render this claim obvious.
2. The method of claim 1, wherein the Downloadable includes an applet.	Signed Java includes applets.
3. The method of claim 1,	Authenticode / ActiveX includes active software controls.

wherein the Downloadable includes an active software control.	
4. The method of claim 1, wherein the Downloadable includes a plugin.	<p>The '780 patent specification provides no understanding of what a plugin might be. If Finjan later defines this term, and it effects my opinion, I will supplement my report.</p> <p>Regardless of the meaning, if a plugin is some other arbitrary executable code which might be used as input to a hash function, I argued above that it would be obvious to one of skill in the art to do so.</p>
5. The method of claim 1, wherein the Downloadable includes HTML code.	As I have argued above, it would be obvious to one of skill in the art to sign other forms of code, including HTML code.
6. The method of claim 1, wherein the Downloadable includes an application program.	Java applets and ActiveX controls are examples of application programs.
7. The method of claim 1, wherein said fetching includes fetching a first software component referenced by the Downloadable.	Regardless of how many software components are fetched, it would be obvious to one of skill in the art that signed code can be performed on software components, whether all at once or individually.
8. The method of claim 1, wherein said fetching includes fetching all software components referenced by the Downloadable.	Regardless of how many software components are fetched, it would be obvious to one of skill in the art that signed code can be performed on software components, whether all at once or individually.
9. A system for generating a Downloadable ID to identify a Downloadable, comprising:	Both Authenticode and Signed Java disclose digital signatures of code, which necessitates computing a hash function over that code, which is an example of the Downloadable ID.
a communications engine for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable; and	<p>Authenticode and Signed Java both include receiving executable code (e.g., ActiveX controls and Java applets).</p> <p>Both ActiveX and Java code include references from one software component to another. This is well known to one of ordinary skill in the art and recognized in the '780 patent itself.</p> <p>In order for a client to receive ActiveX controls or Java applets, it must inherently have a communications subsystem to obtain those controls or applets.</p>
an ID generator coupled to the communications engine that fetches at least one software component identified by the one or more references, and for	It is well known to one of ordinary skill in the art that when ActiveX and Java code include references to software components (e.g., other java class files or ActiveX controls) that those components will be fetched by the client.

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.	<p>Authenticode and Signed Java both disclosed a method of identifying a particular piece or combination of code. In order for Authenticode and Signed Java to generate an identifier to identify the particular code, Authenticode and Signed Java computed a digital signature, which included a performing a hashing function on the code.</p> <p>Because both Authenticode and Signed Java disclose hashing ActiveX controls and Java applets, respectively, it would be obvious to one of ordinary skill in the art that other arbitrary code, be it referenced or not, could also be included in a hashing function to identify the particular set of code hashed.</p> <p>If Finjan takes the position that merely hashing on Java applets or ActiveX controls is novel, Authenticode and Signed Java anticipate that reading.</p> <p>If Finjan takes the position that hashing a Java applet along with other referenced code to generate a single ID is novel, then it is my opinion that Authenticode and Signed Java render this claim obvious.</p>
10. The system of claim 9, wherein the Downloadable includes an applet.	Signed Java includes applets.
11. The system of claim 9, wherein the Downloadable includes an active software control.	Authenticode / ActiveX includes active software controls.
12. The system of claim 9, wherein the Downloadable includes a plugin.	<p>The '780 patent specification provides no understanding of what a plugin might be. If Finjan later defines this term, and it effects my opinion, I will supplement my report.</p> <p>Regardless of the meaning, if a plugin is some other arbitrary executable code which might be used as input to a hash function, I argued above that it would be obvious to one of skill in the art to do so.</p>
13. The system of claim 9, wherein the Downloadable includes HTML code.	As I have argued above, it would be obvious to one of skill in the art to sign other forms of code, including HTML code.
14. The system of claim 9, wherein the Downloadable includes an application program.	Java applets and ActiveX controls are examples of application programs.
15. The system of claim 9, wherein the ID generator fetches a first software component	Regardless of how many software components are fetched, it would be obvious to one of skill in the art that signed code can be performed on software components, whether all at

referenced by the Downloadable.	once or individually.
16. The method of claim 9, wherein the ID generator fetches all software components referenced by the Downloadable.	Regardless of how many software components are fetched, it would be obvious to one of skill in the art that signed code can be performed on software components, whether all at once or individually.
17. A system for generating a Downloadable ID to identify a Downloadable, comprising:	Both Authenticode and Signed Java disclose digital signatures of code, which necessitates computing a hash function over that code, which is an example of the Downloadable ID.
means for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;	Authenticode and Signed Java both include receiving executable code (e.g., ActiveX controls and Java applets). Both ActiveX and Java code include references from one software component to another. This is well known to one of ordinary skill in the art and recognized in the '780 patent itself.
means for fetching at least one software component identified by the one or more references; and	It is well known to one of ordinary skill in the art that when ActiveX and Java code include references to software components (e.g., other java class files or ActiveX controls) that those components will be fetched by the client.
means for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.	Authenticode and Signed Java both disclosed a method of identifying a particular piece or combination of code. In order for Authenticode and Signed Java to generate an identifier to identify the particular code, Authenticode and Signed Java computed a digital signature, which included a performing a hashing function on the code. Because both Authenticode and Signed Java disclose hashing ActiveX controls and Java applets, respectively, it would be obvious to one of ordinary skill in the art that other arbitrary code, be it referenced or not, could also be included in a hashing function to identify the particular set of code hashed. If Finjan takes the position that merely hashing on Java applets or ActiveX controls is novel, Authenticode and Signed Java anticipate that reading. If Finjan takes the position that hashing a Java applet along with other referenced code to generate a single ID is novel, then it is my opinion that Authenticode and Signed Java render this claim obvious.
18. A computer-readable storage medium storing program code for causing a computer to perform the steps of:	ActiveX and Signed Java are embodied in computer-readable storage media storing program code.

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;	<p>Authenticode and Signed Java both include receiving executable code (e.g., ActiveX controls and Java applets).</p> <p>Both ActiveX and Java code include references from one software component to another. This is well known to one of ordinary skill in the art and recognized in the '780 patent itself.</p>
fetching at least one software component identified by the one or more references; and	<p>It is well known to one of ordinary skill in the art that when ActiveX and Java code include references to software components (e.g., other java class files or ActiveX controls) that those components will be fetched by the client.</p>
performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.	<p>Authenticode and Signed Java both disclosed a method of identifying a particular piece or combination of code. In order for Authenticode and Signed Java to generate an identifier to identify the particular code, Authenticode and Signed Java computed a digital signature, which included a performing a hashing function on the code.</p> <p>Because both Authenticode and Signed Java disclose hashing ActiveX controls and Java applets, respectively, it would be obvious to one of ordinary skill in the art that other arbitrary code, be it referenced or not, could also be included in a hashing function to identify the particular set of code hashed.</p> <p>If Finjan takes the position that merely hashing on Java applets or ActiveX controls is novel, Authenticode and Signed Java anticipate that reading.</p> <p>If Finjan takes the position that hashing a Java applet along with other referenced code to generate a single ID is novel, then it is my opinion that Authenticode and Signed Java render this claim obvious.</p>

EXHIBIT L

Claim Language '822	Ji (U.S. Patent 5,983,348)
1. A processor-based method, comprising:	Ji discloses a computer-based method.
receiving downloadable-information;	<p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)" (Ji Abstract) As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information.</p>
determining whether the downloadable-information includes executable code; and	Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or some other form of executable code.
causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,	<p>It's my understanding that Finjan and Secure Computing agreed to a stipulation that "mobile protection code" is "code capable of monitoring or intercepting potentially malicious code."</p> <p>Ji discloses "the present invention is directed to delivering what is referred to as a 'live agent' (e.g., a security monitoring package) along with e.g. an applet that contains suspicious instructions during a network transfer (e.g. downloading to a client),</p>

	<p>the monitoring package being intended to prevent execution of the suspicious instructions" (Ji 3:45-50). Ji's "live agent" is an example of the '822 patent's "mobile protection code" and Ji's "network transfer" is an example of the '822 patent's communication to an information-destination.</p> <p>Incidentally, the term "information-destination" is disputed. Finjan's construction is "client" while Secure Computing's construction is "a device or process that is capable of receiving and initiating or otherwise hosting a mobile code execution." Ji explicitly uses the term "client", meeting Finjan's construction of the term trivially. Ji's client is capable of receiving and initiating or otherwise hosting a mobile code execution, thus also meeting Secure Computing's construction.</p>
<p>wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.</p>	<p>The phrase "downloadable-information characteristic" and "executable code characteristic" never occur anywhere in the '822 patent beyond the patent claims. The phrase "level of downloadable-information characteristic and executable code characteristic correspondence" is in dispute, with Finjan saying this phrase needs no construction and Secure Computing offering a construction that sheds no light on the meaning of the two "characteristic" terms.</p> <p>Given the lack of intrinsic evidence toward the meaning of these "characteristic" terms, we must look to one of skill in the art at the time of the invention. As such, it is my opinion that "downloadable-information characteristics" are the result of various tests that may be conducted on a downloadable (e.g., whether the file name ends in ".class" or whether a file header begins with the hexadecimal string CAFEBAE). From these tests ("detection-indicators"), the system can then draw inferences about what the type of downloadable might be (e.g., a Java class file). These inferences are "executable code characteristics."</p> <p>Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or</p>

	<p>some other form of executable code.</p> <p>Ji discloses two different ways that Java applets might be downloaded: "A Java applet may contain more than one code module, or class file. Heretofore, this disclosure has assumed that all the class files are packed in one JAR file and downloaded once... However, if the class files are not packed together and are downloaded on an as-needed basis during applet execution, ..." (Ji 7:14-20) As such, Ji discloses an example of two downloadable-information characteristics and two detection-indicators that may be used to infer one executable code characteristic.</p>
2. The method of claim 1, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.	<p>Counsel has informed me that Finjan declined Secure Computing's suggestion to define the word "level" as a thresholding or value. To the extent that Finjan believes that this claim does not require a threshold value, then Ji anticipates this claim, since it must necessarily determine whether the downloadable information is code.</p> <p>Thresholds on values are more commonly used in scenarios where an assessment must be made as to whether something is good or bad while the answer might be entirely unclear. (This problem is familiar to users of spam-classification systems.) Ji discloses the use of threshold values: "A simple security policy is to assign different weights to monitored functions and to make sure the security weight of a session does not exceed a present threshold." (Ji 7:56-59)</p> <p>Thus, if one of ordinary skill in the art were presented with the problem of having to make a probabilistic assessment of whether or not a particular data file include executable code, then it would be obvious to one of ordinary skill in the art to use a threshold value technique.</p>
4. A processor-based method, comprising:	Ji discloses processor-based methods.
receiving downloadable-information;	<p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information</p>

	<p>is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)" (Ji Abstract) As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information.</p>
determining whether the downloadable-information includes executable code; and	<p>Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or some other form of executable code.</p>
causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination.	<p>It's my understanding that Finjan and Secure Computing agreed to a stipulation that "mobile protection code" is "code capable of monitoring or intercepting potentially malicious code."</p> <p>Ji discloses "the present invention is directed to delivering what is referred to as a 'live agent' (e.g., a security monitoring package) along with e.g. an applet that contains suspicious instructions during a network transfer (e.g. downloading to a client), the monitoring package being intended to prevent execution of the suspicious instructions" (Ji 3:45-50). Ji's "live agent" is an example of the '822 patent's "mobile protection code" and Ji's "network transfer" is an example of the '822 patent's communication to an information-destination.</p> <p>Incidentally, the term "information-destination" is disputed. Finjan's construction is "client" while</p>

	<p>Secure Computing's construction is "a device or process that is capable of receiving and initiating or otherwise hosting a mobile code execution." Ji explicitly uses the term "client", meeting Finjan's construction of the term trivially. Ji's client is capable of receiving and initiating or otherwise hosting a mobile code execution, thus also meeting Secure Computing's construction.</p> <p>With respect to "forming a sandboxed package including the mobile protection code and the downloadable-information", it's my understanding that both parties have stipulated that a "sandboxed package" is "a protective environment." Ji discloses "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instruction, as described above. The monitor package contains monitoring functions that are delivered from the server to the client web browser with the instrumental applet... It also contains a security policy checker ... to determine whether the applet being scanned violates the security policy, given the monitoring information" (Ji 7:41-49). Ji's "monitor package" is an example of a protective environment, and thus is an example of a sandbox package.</p>
<p>5. The method of claim 4, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.</p>	<p>Counsel has informed me that Finjan has not asserted claim 5. However, Finjan has asserted claim 6 which is a dependent claim of claim 5, making it necessary for me to address the validity of claim 5.</p> <p>Ji discloses: "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instructions." (Ji 7:37-40)</p> <p>Ji elaborates on this: "If an instruction (a suspicious instruction) that calls an insecure function ... is found during the static scanning, a first instruction sequence (pre-filter) is inserted before that instruction" (Ji 5:22-25)</p> <p>Ji's pre-filter causes the monitoring instructions to be executed prior to the suspicious operations. As such, when a suspicious operation is the first operation to be performed by a Java applet, then</p>

	Ji's monitoring instructions will operate prior to any of the original applet's code. This satisfies the claim's temporal requirements.
6. The method of claim 5, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>Ji also discloses "It also contains a security policy checker (supplied by security policy generator component) to determine whether the applet being scanned violates the security policy, given the monitoring information." (Ji 7:45-49). Ji's security policy checker, included as part of Ji's monitor package, illustrates the inclusion of the protection policies according to which the mobile protection code is operable.</p>
8. The method of claim 6, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination.	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>This is an example of protection policies that correspond with at least one of the information-destination and a user of the information destination.</p>
9. A processor-based system, comprising:	Ji discloses a processor-based system.
an information monitor for receiving downloadable-information;	<p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java</p>

	<p>applets or Active X controls)" (Ji Abstract) As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information.</p> <p>With respect to "information monitor", the '822 patent says "Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism. Information monitor 401 can also provide for transferring to storage 404 or other protection engine elements, configuration information including, for example, user MPC, protection policy, interfacing or other configuration information." ('822, 1-10)</p> <p>Ji discloses "The applets or controls ... are conventionally received from e.g. the Internet or an Intranet at a conventional scanner." (Ji 3:19-22) This is an example of "receiving information via ... communication channel."</p>
a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and	<p>Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or some other form of executable code.</p>
a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an	<p>It's my understanding that Finjan and Secure Computing agreed to a stipulation that "mobile protection code" is "code capable of monitoring or intercepting potentially malicious code."</p> <p>Ji discloses "the present invention is directed to delivering what is referred to as a 'live agent' (e.g., a security monitoring package) along with e.g. an applet that contains suspicious instructions during a network transfer (e.g. downloading to a client), the monitoring package being intended to prevent execution of the suspicious instructions" (Ji 3:45-50). Ji's "live agent" is an example of the '822 patent's "mobile protection code" and Ji's "network transfer" is an example of the '822 patent's</p>

inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the downloadable-information includes executable code.

communication to an information-destination.

Incidentally, the term "**information-destination**" is disputed. Finjan's construction is "client" while Secure Computing's construction is "a device or process that is capable of receiving and initiating or otherwise hosting a mobile code execution." Ji explicitly uses the term "client", meeting Finjan's construction of the term trivially. Ji's client is capable of receiving and initiating or otherwise hosting a mobile code execution, thus also meeting Secure Computing's construction.

With respect to "a **packaging engine** ... for causing mobile protection code ... to be communicated", Ji discloses "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instruction, as described above. The monitor package contains monitoring functions that are delivered from the server to the client web browser with the instrumental applet... It also contains a security policy checker ... to determine whether the applet being scanned violates the security policy, given the monitoring information" (Ji 7:41-49). Ji's "instrumenter component" along with Ji's "network transfer" illustrates a packaging engine.

The phrases "**downloadable-information characteristic**", "**downloadable-information analyzers**" and "**executable code characteristic**" never occur anywhere in the '822 patent beyond the patent claims. The phrase "level of downloadable-information characteristic and executable code characteristic correspondence" is in dispute, with Finjan saying this phrase needs no construction and Secure Computing offering a construction that sheds no light on the meaning of the two "characteristic" terms.

Given the lack of intrinsic evidence toward the meaning of these "characteristic" and "analyzer" terms, we must look to one of skill in the art at the time of the invention. As such, it is my opinion that "downloadable-information characteristics" are the result of various tests that may be conducted on a downloadable (e.g., whether the file name ends in ".class" or whether a file header

	<p>begins with the hexadecimal string CAFEBABE). From these tests ("detection-indicators" or "downloadable-information analyzers"), the system can then draw inferences about what the type of downloadable might be (e.g., a Java class file). These inferences are "executable code characteristics." The synthesis of these inferences, then, occurs in the "inspection controller", which ultimately determines whether a particular piece of downloadable-information is actually executable code.</p> <p>Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or some other form of executable code.</p> <p>Ji discloses two different ways that Java applets might be downloaded: "A Java applet may contain more than one code module, or class file. Heretofore, this disclosure has assumed that all the class files are packed in one JAR file and downloaded once... However, if the class files are not packed together and are downloaded on an as-needed basis during applet execution, ..." (Ji 7:14-20) As such, Ji discloses an example of two downloadable-information characteristics and their downloadable-information analyzers as well as two detection-indicators that may be used to infer one executable code characteristic.</p>
12. A processor-based system, comprising:	
an information monitor for receiving downloadable-information;	<p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins,</p>

	<p>downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)" (Ji Abstract) As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information.</p> <p>With respect to "information monitor", the '822 patent says "Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism. Information monitor 401 can also provide for transferring to storage 404 or other protection engine elements, configuration information including, for example, user MPC, protection policy, interfacing or other configuration information." ('822, 1-10)</p> <p>Ji discloses "The applets or controls ... are conventionally received from e.g. the Internet or an Intranet at a conventional scanner." (Ji 3:19-22) This is an example of "receiving information via ... communication channel."</p>
a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and	<p>Ji discloses "an applet scanner that runs e.g. as an HTTP proxy server and does not require any client-side modification" (Ji 3:7-9) In order to operate as an applet scanner, Ji must necessarily be able to determine whether the content passing through the proxy server is or is not an applet or some other form of executable code.</p>
a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, wherein the	<p>It's my understanding that Finjan and Secure Computing agreed to a stipulation that "mobile protection code" is "code capable of monitoring or intercepting potentially malicious code."</p> <p>Ji discloses "the present invention is directed to delivering what is referred to as a 'live agent' (e.g., a security monitoring package) along with e.g. an</p>

packaging engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC generator for forming a sandbox package including the MPC and the downloadable-information, and a transfer engine for causing the sandbox package to be communicated to the at least one information-destination.

applet that contains suspicious instructions during a network transfer (e.g. downloading to a client), the monitoring package being intended to prevent execution of the suspicious instructions" (Ji 3:45-50). Ji's "live agent" is an example of the '822 patent's "mobile protection code" and Ji's "network transfer" is an example of the '822 patent's communication to an information-destination.

Incidentally, the term "**information-destination**" is disputed. Finjan's construction is "client" while Secure Computing's construction is "a device or process that is capable of receiving and initiating or otherwise hosting a mobile code execution." Ji explicitly uses the term "client", meeting Finjan's construction of the term trivially. Ji's client is capable of receiving and initiating or otherwise hosting a mobile code execution, thus also meeting Secure Computing's construction.

With respect to "a **packaging engine** [that] comprises an MPC generator ... a linking engine ... and a transfer engine", Ji discloses "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instruction, as described above. The monitor package contains monitoring functions that are delivered from the server to the client web browser with the instrumental applet... It also contains a security policy checker ... to determine whether the applet being scanned violates the security policy, given the monitoring information" (Ji 7:41-49). Ji's "instrumenter component" along with Ji's "network transfer" illustrates a packaging engine comprising an **MPC generator**, which includes a **linking engine** to form a **sandbox package** and send it to the client with a **transfer engine**.

13. The system of claim 12, wherein the packaging engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.

Ji discloses "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instruction, as described above. The monitor package contains monitoring functions that are delivered from the server to the client web browser with the instrumental applet... It also contains a security policy checker ... to determine whether the applet being scanned violates the security policy, given

	<p>the monitoring information" (Ji 7:41-49).</p> <p>Ji's instrumenter component satisfies the limitations of this claim. In particular, Ji's security policy checker illustrates protection policies according to which the MPC is operable.</p>
<p>14. The system of claim 13, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.</p>	<p>Counsel has informed me that Finjan has not asserted claim 14. However, Finjan has asserted claim 15 which is a dependent claim of claim 14 making it necessary for me to address the validity of claim 14.</p> <p>Ji discloses: "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instructions." (Ji 7:37-40)</p> <p>Ji elaborates on this: "If an instruction (a suspicious instruction) that calls an insecure function ... is found during the static scanning, a first instruction sequence (pre-filter) is inserted before that instruction" (Ji 5:22-25)</p> <p>Ji's pre-filter causes the monitoring instructions to be executed prior to the suspicious operations. As such, when a suspicious operation is the first operation to be performed by a Java applet, then Ji's monitoring instructions will operate prior to any of the original applet's code. This satisfies the claim's temporal requirements.</p>
<p>15. The system of claim 14, wherein the protection policies correspond with policies of at least one of the information-destination and a user of the information destination.</p>	<p>I interpret this claim to refer to client-based policies ("policies of ... the information-destination") and user-based policies ("policies of ... a user of the information destination").</p> <p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>As such, Ji illustrates the wide variety of available policies, satisfying this limitation.</p>
<p>16. A processor-based method, comprising:</p>	<p>Ji discloses a processor-based method.</p>
<p>receiving, at an information re-communicator, downloadable-information,</p>	<p>My understanding is that Finjan and Secure Computing dispute the definition of an</p>

<p>including executable code; and</p>	<p>"information re-communicator." Finjan proposes that it means "server" while Secure Computing proposes that it means "information supplier or intermediary for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes."</p> <p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)... Static scanning at the HTTP proxy server identifies suspicious instructions and instruments them." (Ji Abstract)</p> <p>Ji also discloses in claims 11, 26, 31, and 33 the concept of a network scanner running on a server computer.</p> <p>As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information. Likewise, Ji discloses the information re-communicator, whether under Secure Computing's construction or Finjan's construction of information re-communicator.</p>
<p>causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, wherein the causing is</p>	<p>According to the '822 patent, a mobile code executor can be "an OS task manager, Java Virtual Machine ('JVM'), etc." ('822, 8:42-43)</p> <p>Ji discloses the Java Virtual Machine, one of the examples of a mobile code executor: "After the code of an applet is downloaded, e.g. via the</p>

accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the web browser (client) running on that local computer." (Ji 4:30-35)

It's my understanding that Finjan and Secure Computing agreed to a stipulation that "mobile protection code" is "code capable of monitoring or intercepting potentially malicious code."

Ji discloses "the present invention is directed to delivering what is referred to as a 'live agent' (e.g., a security monitoring package) along with e.g. an applet that contains suspicious instructions during a network transfer (e.g. downloading to a client), the monitoring package being intended to prevent execution of the suspicious instructions" (Ji 3:45-50). Ji's "live agent" is an example of the '822 patent's "mobile protection code" and Ji's "network transfer" is an example of the '822 patent's communication to an information-destination.

Incidentally, the term "information-destination" is disputed. Finjan's construction is "client" while Secure Computing's construction is "a device or process that is capable of receiving and initiating or otherwise hosting a mobile code execution." Ji explicitly uses the term "client", meeting Finjan's construction of the term trivially. Ji's client is capable of receiving and initiating or otherwise hosting a mobile code execution, thus also meeting Secure Computing's construction.

With respect to "forming a sandboxed package including the mobile protection code and the downloadable-information", it's my understanding that both parties have stipulated that a "sandboxed package" is "a protective environment." Ji discloses "The Java instrumenter component instruments the Java class files, e.g. by inserting monitoring instructions (e.g. pre and post filter calls) before and after each suspicious instruction, as described above. The monitor package contains monitoring functions that are delivered from the server to the client web browser with the instrumental applet... It also contains a security policy checker ... to determine whether the applet being scanned violates the security policy, given the monitoring information" (Ji 7:41-49). Ji's "monitor package" is an example of a protective

	environment, and thus is an example of a sandbox package.
17. The method of claim 16, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>Ji also discloses "It also contains a security policy checker (supplied by security policy generator component) to determine whether the applet being scanned violates the security policy, given the monitoring information." (Ji 7:45-49). Ji's security policy checker, included as part of Ji's monitor package, illustrates the inclusion of the protection policies according to which the mobile protection code is conducted.</p>
18. A sandboxed package formed according to the method of claim 17.	I do not understand how claim 18 is in any way different than claim 17. I reserve the right to supplement this in response to any distinctions made later by Finjan.
19. The method of claim 17, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information.	<p>Counsel has informed me that Finjan has not asserted claim 19. However, Finjan has asserted claim 20 which is a dependent claim of claim 19, making it necessary for me to address the validity of claim 19.</p> <p>I do not understand how claim 19 is in any way different than claim 17. I reserve the right to supplement this in response to any distinctions made later by Finjan.</p>
20. The method of claim 19, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>These are examples of destination-characteristics of the destination.</p>
21. The method of claim 20, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>Ji's user is an example of the destination user. Ji's clients are examples of the destination devices. Ji's applets are examples of the destination processes.</p>
22. A sandboxed package formed according to the method of claim 16.	I do not understand how claim 22 is in any way different from claim 16. I reserve the right to supplement this in response to any distinctions

	made later by Finjan.
24. The method of claim 16, wherein the re-communicator is at least one of a firewall and a network server.	Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)... Static scanning at the HTTP proxy server identifies suspicious instructions and instruments them." (Ji Abstract) Ji's HTTP proxy server is a network server.
26. The method of claim 25, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.	<p>This claim uses the term "concatenation" which has a specific meaning to a computer scientists, namely directly appending one string after the end of another string.</p> <p>One of ordinary skill in the art would know that this form of concatenation would not function with Java applets, JavaScript scripts, ActiveX objects, or any other mobile code systems in widespread use.</p> <p>The Oxford English Dictionary offers this definition of concatenation: "Union by chaining or linking together." This could be interpreted broadly as "combining." Ji discloses: "The instrumented applet is then transferred to the client (web browser) together with security monitoring code." (Ji Abstract) Ji also discloses: "The pre and post filter and monitoring package security policy functions are combined with the instrumented applet code in a single JAR (Java archive) file format at the server and downloaded to the web browser." (Ji 6:38-41) Under this broader definition of concatenation, the Ji disclosure is an example of forming a sandboxed package by concatenating mobile protection code, a policy, and a downloadable.</p>
27. The method of claim 16, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.	<p>Ji discloses: "the suspicious instructions each may (or may not) be instrumented as described above; the instrumentation involves altering suspicious instruction such as by adding code ... or altering the suspicious instructions by replacing any suspicious instructions with other instructions." (Ji 3:50-56)</p> <p>Ji further discloses: "First, one can 'trick' applets into calling particular functions supplied by the scanner during the dynamic linking stage. This is</p>

	<p>done by replacing the browser Java library routines with the scanner's monitoring routines of the same name." (Ji 4:8-12)</p> <p>Ji's replacement of instructions is an example of modifying the downloadable interfaces. Ji's replacement of the browser routines with the scanner's routines is an example of diverting attempted operations to the mobile protection code.</p>
28. A processor-based system, comprising:	Ji discloses a processor-based system.
receiving means for receiving, at an information re-communicator, downloadable-information, including executable code; and	<p>My understanding is that Finjan and Secure Computing dispute the definition of an "information re-communicator." Finjan proposes that it means "server" while Secure Computing proposes that it means "information supplier or intermediary for servicing one or more further interconnected evices or porcesses or interconnected levels of devices or processes."</p> <p>According to Secure Computing's construction, downloadable-information is "data downloaded from one computer to another."</p> <p>Finjan's construction of downloadable-information is "program or document that can contain mobile code."</p> <p>Following Finjan's construction, mobile code is still an undefined term, but the '822 patent discloses "Java applets, ActiveX controls, JavaScript scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other 'Downloadables' or 'mobile code' in whole or in part." ('822 Abstract) As such, "mobile code" seems to include, at least, Java applets or ActiveX controls.</p> <p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)... Static scanning at the HTTP proxy server identifies suspicious instructions and instruments them." (Ji Abstract)</p> <p>Ji also discloses in claims 11, 26, 31, and 33 the concept of a network scanner running on a server</p>

	<p>computer.</p> <p>As such, Ji discloses receiving downloadable-information, whether under Secure Computing's construction or Finjan's construction of downloadable-information. Likewise, Ji discloses the information re-communicator, whether under Secure Computing's construction or Finjan's construction of information re-communicator.</p>
<p>mobile code means communicatively coupled to the receiving means for causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.</p>	<p>I understand that Secure Computing has argued that "mobile code means" is a means-plus-function claim which lacks structure in the specification and is thus indefinite.</p> <p>Grammatically, the body of this claim element is the function of the mobile code means. My understanding is that the function performed by the mobile code means is "<i>causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.</i>" (italics added)</p> <p>Finjan's answering brief refers to disclosed structure of an information destination. The structure to which Finjan refers does not include any description of (1) means for processing operations by mobile protection code, (2) forming a sandbox package, and (3) delivering the sandbox package to the information-destination. All three of these functions are required functions to be performed by the mobile code means. Moreover, counsel has informed me that, as a matter of law, two different terms within the same claim cannot mean the same thing. Consequently, "mobile code means" cannot be synonymous with an "information destination" or "downloadable-information destination", as suggested by Finjan.</p> <p>In my review of the '822 patent, I was unable to locate a structure that performs the three functions identified in the previous paragraph.</p>

	<p>Without an understanding of the specific structure required by the limitation, all I can do is point to the functionality disclosed in Ji that corresponds to the functionality provided by the mobile code means. Please see my analysis of the last element of claim 16 which is a verbatim recital of the functional limitations expressed in this claim element.</p>
<p>29. The system of claim 28, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.</p>	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53)</p> <p>Ji also discloses "It also contains a security policy checker (supplied by security policy generator component) to determine whether the applet being scanned violates the security policy, given the monitoring information." (Ji 7:45-49). Ji's security policy checker, included as part of Ji's monitor package, illustrates the inclusion of the protection policies according to which the mobile protection code is conducted.</p>
<p>30. The system of claim 29, wherein the forming comprises generating the mobile protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.</p>	<p>I do not understand how claim 30 is in any way different than claim 28. I reserve the right to supplement this in response to any distinctions made later by Finjan.</p>
<p>31. The system of claim 30, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.</p>	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53) These are examples of destination-characteristics of the destination.</p>
<p>32. The system of claim 31, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.</p>	<p>Ji discloses "Different clients, users, and applets may have different security policies." (Ji 7:52-53) Ji's user is an example of the destination user. Ji's clients are examples of the destination devices. Ji's applets are examples of the destination processes.</p>
<p>33. The system of claim 28, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.</p>	<p>Counsel has informed me that Finjan has not asserted claim 33. However, Finjan has asserted claim 34 which is a dependent claim of claim 33, making it necessary for me to address the validity of claim 33.</p>

	<p>All modern computer operating systems include buffering of networked transmission. As such, any installation of the system on a modern computer with a modern operating system will satisfy this limitation.</p>
<p>34. The system of claim 33, wherein the re-communicator is at least one of a firewall and a network server.</p>	<p>Ji discloses "a network scanner for security checking of application programs (e.g. Java applets or Active X controls)... Static scanning at the HTTP proxy server identifies suspicious instructions and instruments them." (Ji Abstract) Ji's HTTP proxy server is a network server.</p>
<p>35. The system of claim 34, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.</p>	<p>Ji discloses: "the suspicious instructions each may (or may not) be instrumented as described above; the instrumentation involves altering suspicious instruction such as by adding code ... or altering the suspicious instructions by replacing any suspicious instructions with other instructions." (Ji 3:50-56)</p> <p>Ji further discloses: "First, one can 'trick' applets into calling particular functions supplied by the scanner during the dynamic linking stage. This is done by replacing the browser Java library routines with the scanner's monitoring routines of the same name." (Ji 4:8-12)</p> <p>Ji's replacement of instructions is an example of modifying the downloadable interfaces. Ji's replacement of the browser routines with the scanner's routines is an example of diverting attempted operations to the mobile protection code.</p>

EXHIBIT M

<p>'361 Patent</p> <p>Claim Language</p>	<p>Finjan Vital Security product</p>
<p>1. A system for authorizing client access to a network resource, comprising:</p>	<p>"Finjan offers secure web gateway products for the enterprise market." (www.finjan.com/Content.aspx?id=190)</p>
<p>a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and</p>	<p>"To automatically import LDAP users from the Active Directory: 1. Enable the Import LDAP Users option, ..." (Vital Security User Guide, FIN008796)</p> <p>"To manually import LDAP users from the active Directory: ... 3. Select the required directory, and then, from the list of Available Groups, select the groups that you want to import ..." (Vital Security User Guide, FIN008796)</p> <p>LDAP is a network protocol for accessing directories of information concerning an entity's organization, including its individual users and the groups to which they might belong.</p> <p>Furthermore, I directly observed the results of sniffing the packets between a Finjan Vital Security product and a Microsoft ActiveDirectory system, running on separate computers where the Vital Security product was configured to use the ActiveDirectory system. The captured packets indicated that protocol spoken between these two systems was LDAP.</p>
<p>a firewall that is configured to intercept network resource requests from a plurality of client users on an internal network,</p>	<p>Finjan's Vital Security product is a firewall.</p> <p>"Vital Security can be deployed as a transparent HTTP proxy, in conjunction with a third-party content switch, or [as] a layer-4 router in the network" (Vital Security User Guide, FIN008740). In any of these configurations, internal computers read documents from the external network with their requests received by and handled by the Vital Security product. Furthermore, I witnessed a Vital Security system which received network resource requests (i.e., attempts to load a web page) from an "internal" client user machine.</p> <p>Furthermore, I read portions of the deposition transcript of Mr. Frommer, where he indicated that the Vital Security product can intercept both network requests and responses (Frommer Depo. rough, 117-120).</p>

<p>said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter,</p>	<p>Finjan's Vital Security product is a firewall.</p> <p>"As soon as Vital Security authenticates a user by confirming the matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found." (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is "user name". The least specific identification listed is "LDAP group". (Groups and roles are synonymous.)</p> <p>Based on my investigation of the product, the Vital Security product downloads the contents of an LDAP directory and stores it locally. The contents will be, for example, user names and users' assigned groups.</p> <p>Enforcing the assigned policy, as described in the above quote from the Management Console Reference Guide, is an example of the actions of an authorization filter, which will make comparisons of the user/group database, originally from the LDAP server, with the policy rules that can be set by the Vital Security administrator.</p>
<p>wherein said authorization filter is generated based on a directory schema that is predefined by said entity.</p>	<p>The management console allows authorization rules to be associated with users and groups that come from the LDAP directory. These authorization rules are "generated based on" the LDAP directory. Consequently, this is an example of the authorization filter required by this claim element.</p>
<p>2. The system of claim 1, wherein said at least one directory is a lightweight directory access protocol directory.</p>	<p>"To automatically import LDAP users from the Active Directory: 1. Enable the Import LDAP Users option, ..." (Vital Security User Guide, FIN008796)</p> <p>"To manually import LDAP users from the active Directory: ... 3. Select the required directory, and then, from the list of Available Groups, select the groups that you want to import ..." (Vital Security User Guide, FIN008796)</p> <p>LDAP is the Lightweight Directory Access Protocol.</p>
<p>3. The system of claim 1, wherein said authorization filter is specified using a graphical user interface.</p>	<p>The Vital Security User Guide describes the graphical user interface used by Finjan's Vital Security products, including the user interfaces for specifying authorization filters ("policies"). See, for example, Vital Security User Guide, FIN008782.</p> <p>Furthermore, I have witnessed this user interface and used it to</p>

	specify policies.
4. The system of claim 1, wherein said authorization filter implements a per-user authentication scheme.	<p>Vital Security supports per-user authentication:</p> <p>“As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found.” (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is “user name”.</p>
5. The system of claim 1, wherein said authorization filter implements a per-service authentication scheme.	<p>Vital Security supports “URL lists” which allow rules to be implemented on a per-service basis. (Vital Security User Guide, FIN008779), allowing some users access to a given service while denying access to other users.</p>
7. The system of claim 1, wherein said firewall is configured to query multiple directories.	<p>“In the LDAP tab (Figure 103), you can add or delete LDAP directories and edit the directory user information.” (Vital Security User Guide, FIN008794)</p>
8. An authentication method at a firewall, comprising the steps of:	<p>Finjan's Vital Security product is a firewall.</p> <p>Vital Security can authenticate users and/or user groups:</p> <p>“As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found.” (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is “user name”.</p>
(a) receiving a network resource request from a client user at an internal network;	<p>“Vital Security can be deployed as a transparent HTTP proxy, in conjunction with a third-party content switch, or [as] a layer-4 router in the network” (Vital Security User Guide, FIN008740). In any of these configurations, internal computers read documents from the external network with their requests received by and handled by the Vital Security product. Furthermore, I witnessed a Vital Security system which received network resource requests (i.e., attempts to load a web page) from an “internal” client user machine.</p>
(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is	<p>Vital Security allows policies (“authorization filters”) to be expressed in terms of user groups (which are maintained by LDAP directory services).</p>

<p>based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;</p>	<p>"To automatically import LDAP users from the Active Directory: 1. Enable the Import LDAP Users option, ..." (Vital Security User Guide, FIN008796)</p> <p>"To manually import LDAP users from the active Directory: ... 3. Select the required directory, and then, from the list of Available Groups, select the groups that you want to import ..." (Vital Security User Guide, FIN008796)</p> <p>LDAP is a network protocol for accessing directories of information concerning an entity's organization. The directory schema, user groupings, and so forth for the LDAP server can be defined by an organization.</p> <p>The query to the LDAP server is based on the authorization filter in that the query downloads the necessary information (users, groups) from the LDAP server to satisfy the authorization filter's needs.</p>
<p>(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and</p>	<p>"As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found." (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is "user name". "LDAP Group" is also listed.</p>
<p>(d) permitting said network resource request through said firewall if said authorization filter is satisfied.</p>	<p>A Vital Security policy ("authorization filter") can specify that a particular network resource request be either allowed or denied. Furthermore, I have witnessed a Vital Security appliance in use, where we changed the policy and observed requests being allowed or denied, depending on whether the policy was satisfied.</p>
<p>9. The method of claim 8, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.</p>	<p>Vital Security supports LDAP servers, as described above.</p>
<p>10. The method of claim 8, further comprising the step of specifying an authorization filter using a graphical user interface.</p>	<p>Vital Security has graphical user interface support for specifying its policies, as described above.</p>
<p>11. The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that</p>	<p>"As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found." (Vital Security User Guide,</p>

implements a per-user authentication scheme.	FIN008750-FIN008751) The most specific identification listed is "user name". Furthermore, I have witnessed the Vital Security product's user interface and used it to specify per-user policies.
12. The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.	<p>Vital Security supports "URL lists" which allow rules to be implemented on a per-service basis. (Vital Security User Guide, FIN008779), allowing some users access to a given service while denying access to other users.</p> <p>Vital Security also supports "black and white lists" as part of its specification of per-service authentication schemes (Vital Security User Guide, FIN008774).</p>
14. The method of claim 8, wherein step (b) comprises the step of querying multiple directories.	"In the LDAP tab (Figure 103), you can add or delete LDAP directories and edit the directory user information." (Vital Security User Guide, FIN008794)
15. A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:	<p>Finjan Vital Security products are built using standard computer system parts, including processors, and support user and group authentication.</p> <p>"As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found." (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is "user name". "LDAP Group" is also listed.</p>
a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:	Finjan projects are built using standard computer system techniques, including readable program code executing on a computer system.
first computer readable program code for enabling the computer system to receive a network resource request from a client user at an internal network;	"Vital Security can be deployed as a transparent HTTP proxy, in conjunction with a third-party content switch, or [as] a layer-4 router in the network" (Vital Security User Guide, FIN008740). In any of these configurations, internal computers read documents from the external network with their requests received by and handled by the Vital Security product. Furthermore, I witnessed a Vital Security system which received network resource requests (i.e., attempts to load a web page) from an "internal" client user machine.

<p>second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;</p>	<p>Vital Security allows policies ("authorization filters") to be expressed in terms of user groups (which are maintained by LDAP directory services).</p> <p>"To automatically import LDAP users from the Active Directory: 1. Enable the Import LDAP Users option, ..." (Vital Security User Guide, FIN008796)</p> <p>"To manually import LDAP users from the active Directory: ... 3. Select the required directory, and then, from the list of Available Groups, select the groups that you want to import ..." (Vital Security User Guide, FIN008796)</p> <p>LDAP is a network protocol for accessing directories of information concerning an entity's organization.</p>
<p>third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and</p>	<p>"As soon as Vital Security authenticates a user by confirming a matching identifier, the assigned policy is enforced. The identification parameters are checked from the more specific to the less specific ... until a match is found." (Vital Security User Guide, FIN008750-FIN008751) The most specific identification listed is "user name". "LDAP Group" is also listed.</p>
<p>fourth computer readable program code for enabling the computer system to permit said network resource request through a firewall if said authorization filter is satisfied.</p>	<p>As above, Finjan's "enforcing" of the policy, if the resource request is authorized, is simply the action of satisfying the request.</p>

EXHIBIT N

'010 Patent Claim Language	<p>Finjan "Vital Security for Enterprise Documents" product</p> <p>It is my understanding that "Vital Security for Documents" and "Vital Security for Enterprise Documents" are the same product. As such, I will refer to both as "Vital Security for Documents."</p> <p>Referenced documents:</p> <ol style="list-style-type: none"> 1. "Vital Security for Enterprise Documents," Copyright 2003, two page flyer, hereafter "Vital Flyer." 2. "Finjan : Vital Security for Documents," marketing material found at URL below, hereafter "Presence". http://www.presence-security.co.uk/index.cfm/page/products.details.cfm/id/196 3. "Finjan Software Launches a New Product - Vital Security for SSL", press release dated July 19, 2004 at the URL below. Hereafter "PR". http://www.finjan.com/Pressrelease.aspx?PressLan=329&id=424&lan=3
37. A computer-readable medium comprising program code, in a system having an internal network and an interface to an external network, for handling requests from the external network for documents stored on the internal network, the program code comprising:	Vital Security for Documents supports requests from internal and external networks for documents: "Vital Security™ for Documents: Enables companies to control the access, authorization and distribution of sensitive documents internally and externally. It allows trusted users to view critical business information and intellectual property unimpeded, while preventing those properties from being digitally distributed, electronically copied, or physically replicated." (PR)
program code for defining a plurality of users, including a first and a second user;	Vital Security for Documents allows a plurality of users to be defined: "Create unique policies for individuals or groups of users and creates consistent policy implementation for all users." (Presence)
program code for assigning each user to one or more roles, wherein assigning includes assigning the first user to a first role and the second user to a second role;	Vital Security for Documents allows users to be assigned to roles: "Granular policies and roles-based rules determine who has access to confidential documents and what actions they have the rights to perform." (Presence)
program code for defining	Vital Security for Documents allows roles to be assigned to users:

documents accessible to the users, wherein defining includes limiting access to documents as a function of the roles assigned to the user;	<p>"Granular policies and roles-based rules determine who has access to confidential documents and what actions they have the rights to perform." (Presence)</p> <p>Vital Security for Documents allows limited access to documents to be defined as a function of the rules assigned to the roles assigned to the user: "System administrators have the ability to control and set policies for copying, printing, saving, forwarding and screen capturing business documents for individual users or user groups." (Presence)</p>
program code for receiving a document request from the external network;	Vital Security for Documents can receive document requests from an external network: "Vital Security™ for Documents: Enables companies to control the access, authorization and distribution of sensitive documents internally and externally. It allows trusted users to view critical business information and intellectual property unimpeded, while preventing those properties from being digitally distributed, electronically copied, or physically replicated." (PR)
program code for determining a user and a role associated with the document request;	<p>Vital Security for Documents can determine a user and a role associated with a document request: "Create unique policies for individuals or groups of users and creates consistent policy implementation for all users." (Presence)</p> <p>"Granular policies and roles-based rules determine who has access to confidential documents and what actions they have the rights to perform." (Presence)</p>
program code for authenticating the user associated with the document request;	<p>Vital Security for Documents can authenticate the user associated with a document request: "It uses standard keys and key exchange protocols to encrypt documents and authenticate users." (Vital Flyer, SC 11258)</p> <p>"Because keys necessary to decrypt protected documents can be retrieved only after the user successfully authenticates with a Vital Security Key Server, a company using Finjan's Vital Security is assured that protected intellectual property will not be usable outside of the company, even if Vital Security is installed at both locations." (Vital Flyer, SC 11259)</p>
program code for determining if users in the role associated with the document request have permission to access the document requested; and	Vital Security for Documents can determine if users in the role associated with the document request have permission to access the document requested: "Granular policies and roles-based rules determine who has access to confidential documents and what actions they have the rights to perform." (Presence)

	<p>"Create unique policies for individuals or groups of users and creates consistent policy implementation for all users." (Presence)</p>
<p>program code for, if users in the role associated with the document request have permission to access the document requested, retrieving the document requested from the internal network and delivering the document to the user associated with the document request.</p>	<p>Vital Security for Documents can determine if users in the role associated with the document request have permission to access the document requested (see claim element above).</p> <p>Vital Security for Documents can retrieve the document requested from the internal network and deliver the document to the user associated with the document request: "Vital Security Server Integrates with Web servers to intercept requests for documents, and encrypts them before they are served." (Vital Flyer, SC 11259)</p> <p>"The Vital Security Server integrates with corporate Web servers and unobtrusively monitors requests for protected information. When a request is received, the Vital Security Server intercepts it, retrieves the information and encrypts it as it is being served to the end-user's workstation." (Vital Flyer, SC 11259)</p> <p>Based on these quotes, the Vital Security for Documents server intercepts requests and retrieves documents from an internal Web server. There are three possible ways in which the Vital Security for Documents Server might integrate with an internal web server: running on a separate computer and connected via a network, running on the same computer and connected via a "loopback" (i.e., simulated) network, or running as a plug-in that attaches to the web server directly.</p> <p>In the first case, running on a separate computer, then the Vital Security for Documents server is clearly receiving requests for documents and communicating them on the internal network to the web server.</p> <p>In the second case, running on the same computer, a "loopback" network connection would be used for the Vital Security for Documents server to make requests of the web server. This connection is entirely internal to the computer but is still clearly a network connection.</p> <p>The third case appears unlikely given the disclosures available about the Vital Security for Documents product. If it were designed as a web server plug-in, then it would be far more specialized in the servers that it would integrate with, whereas the flyer states that Vital Security for Documents supports "Microsoft IIS4, IIS5", "iPlanet 4.0, 4.1", and "Apache 1.3" (Vital Flyer, SC 11259), which represent three completely distinct web server products with incompatible plug-in architectures. Furthermore, if Vital Security for Documents was fully</p>

integrated as a server plugin, the marketing materials would presumably describe the benefits that could be achieved with such an integration (e.g., integrating with those web servers' own access control and logging mechanisms). No such benefits are described. From this, I can only conclude that Vital Security for Documents was not designed to operate as a web server plug-in.

EXHIBIT 2, PART 1

ROBINS, KAPLAN, MILLER & CIRESI LLP

2800 LASALLE PLAZA
800 LASALLE AVENUE
MINNEAPOLIS, MN 55402-2015
TEL: 612-349-8500 FAX: 612-339-4181
www.rkmc.com

ATTORNEYS AT LAW

TREVOR J. FOSTER
612-349-0859
TJFoster@rkmc.com

November 30, 2007

VIA FEDERAL EXPRESS & E-MAIL

James Hannah, Esq.
Perkins Coie LLP
101 Jefferson Drive
Menlo Park, CA 94025-1114

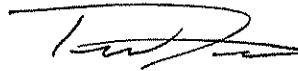
Re: *Finjan Software, Ltd. v. Secure Computing Corp., et al.*
Court File No.: 06-369 GMS
Our File No.: 028487.0002

Dear James:

Please find attached Dr. Wallach's supplementary charts. Theses are the charts corresponding with exhibits C-J. These charts expressly apply Dr. Wallach's analysis from claim 1 and its dependent claims in the '194 Patent to the parallel limitations in claims 32-36 and 65 of the '194 Patent by copying the appropriate text and placing it next to the appropriate limitations. Dr. Wallach has not provided any new substantive analysis in this supplement that was not in his original report.

Best regards,

ROBINS, KAPLAN, MILLER & CIRESI L.L.P.



Trevor J. Foster

TJF/dgt

EXHIBIT C

Claim Language '194 Patent	Shaio (U.S. Patent No. 6,571,338)
1. A computer-based method, comprising the steps of:	Shaio discusses the use of computers.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	<p>Shaio discloses an "intelligent firewall that provides real-time security testing of network packets, which may include executable code, such as applets" (2:25-27)</p> <p>Under Secure Computing's proposed construction of the terms in the '194 patent, a Downloadable is a program or document containing an executable application program that can be downloaded from one computer to another computer. Shaio's "network packets, which may include executable code" meet this definition.</p> <p>Under Finjan's proposed construction, a Downloadable is a "program or document containing mobile code." Shaio's network packets, which may include executable code" also meet this definition.</p> <p>Shaio discusses network packets, which include the network address of the destination to which the packets are being transmitted, i.e., the client computer (see, e.g., Shaio 5:1-5). This satisfies "addressed to a client" under both Finjan's and Secure Computing's proposed constructions.</p> <p>Under Secure Computing's proposed construction of a "server that serves as a gateway to the client," there must be "a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client." Shaio discloses "SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch." (3:55-57) Shaio's Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p> <p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no construction, saying that the server is "simply 'an intermediate between the Internet and the user.'" With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio's firewall meets this</p>

	limitation.
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	<p>Shaio discloses “additional security may be provided by intelligent firewall... For example, a bytecode verifier may parse the executable code portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses.” (5:7-11)</p> <p>The ‘194 patent explains the concepts of security profile data and suspicious computer operations, saying “it will be appreciated that the code scanner ... may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.” (5:54-57) Shaio’s Java bytecode verifier takes a number of steps, detailed in U.S. patent 5,740,441, incorporated by reference. These steps include, for example, looking at the list of method call operations and field references in the bytecode and verifying that they are properly formed (i.e., they point to a valid class or method which is legal to reference — this is a security policy being enforced by the Java bytecode verifier).</p> <p>In this scenario, the list of bytecodes being verified by Shaio are an example of the Downloadable security profile data in the ‘194 patent. The rules being evaluated by the bytecode verifier are examples of the security policy in the ‘194 patent. As a result, Shaio’s bytecode verifier meets the limitations of this claim element.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	Shaio discloses “there is a need for an intelligent firewall that provides real-time security testing of network packets, which may include executable code such as applets, and determines the risk level, i.e., trust worthiness, of each packet before permitting a lower-risk subset of the network packets to execute on anyone of the secure computers” (2:25-31) which meets this limitation of the ‘194 patent.
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	Shaio discloses, “For example, a bytecode verifier may parse the executable portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses” (5:8-11) which meets this limitation of the ‘194 patent.
3. The method of claim 2, wherein the security policy includes an access	The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to

control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Shaio’s Java bytecode verifier, as described in U.S. patent 5,740,441, incorporated by reference, discusses the validation of method calls and field references. Shaio describes a list of criteria that are tested against Java bytecode files (18:49-54) including checking that class names, field and method references, and so forth have valid names. This structure in Shaio is an example of an access control list to which the Downloadable security profile data is compared.</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft’s Authenticode and Sun’s Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	<p>A URL (uniform resource locator) is a string describing the location and name of content which may be downloaded. URLs typically take the form of http://somehost.com/some/name/, where “somehost.com” is a network address and “some/name/” is a file available from that server.</p> <p>Shaio describes a firewall which makes determinations based on the “source address of an executable packet” (see, e.g., 2:64-65). A URL is common way to express a source address. As such, Shaio satisfies this element of the ‘194 patent.</p>
6. The method of claim 5, wherein the known URL is a trusted URL.	<p>Shaio discloses, “a source/destination network address is considered uncertain if there is no match between the network address and a list of pre-approved secured network addresses” (4:35-38).</p> <p>Shaio discloses “if the firewall determines within the first degree of certainty that the source address is associated with anyone of the secured nodes, then the firewall permits the executable packet to proceed to the secured computer.” (3:3-6)</p> <p>URLs can describe the addresses of both the source and the secured nodes. As such, Shaio satisfies this</p>

	element of the '194 patent.
7. The method of claim 5, wherein the known URL is an untrusted URL.	Shaio discloses "Conversely, if firewall ... is uncertain or determines that the source address of the packet is outside ... then the packet is rejected." (5:1-5) Again, URLs can describe the addresses. As such, Shaio satisfies this element of the '194 patent.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	Shaio discloses "the present invention provides a method and apparatus for determining the trust worthiness of executable packets, e.g., internet applets, being transmitted within a computer network." Shaio satisfies this element of the '194 patent.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	As above, Shaio discusses "executable packets, e.g., internet applets." ActiveX is another example of an executable packet. Shaio satisfies this element of the '194 patent.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	As above, Shaio discusses "executable packets, e.g., internet applets." JavaScript scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	As above, Shaio discusses "executable packets, e.g., internet applets." Visual Basic scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Shaio's bytecode verifier is applied to all executable packets, regardless of origin or destination. Shaio satisfies this element of the '194 patent.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	<p>Shaio discloses, "a source/destination network address is considered uncertain if there is no match between the network address and a list of pre-approved secured network addresses" (4:35-38).</p> <p>The presence of this uncertainty, in Shaio, allows for different security policies to be applied. Shaio thusly satisfies this element of the '194 patent.</p>
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	Shaio discloses "network addresses may include a prefix field and a machine field, with the prefix field identifying clusters of computer systems couples to the respective network nodes, and the machine field identifying computer systems within each cluster." (4:39-43) This mechanism allows machines to be grouped together for the purpose of expressing a security policy. Thus, Shaio satisfies this element of the '194 patent.
24. The method of claim 1, further	This limitation describes the behavior of signature-

<p>comprising the step of comparing the Downloadable against a known Downloadable.</p>	<p>based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)</p>
<p>25. The method of claim 24, wherein the known Downloadable is hostile.</p>	<p>This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)</p>
<p>26. The method of claim 24, wherein the known Downloadable is non-hostile.</p>	<p>One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and it would be obvious for any other firewall to adopt the same features.</p> <p>To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of</p>

	the program.” (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	<p>Hershey discloses a computer system that “provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity.” (5:23-27)</p> <p>Hershey and Shaio both disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from both systems together in a single firewall solution.</p>
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has “whitelist” and “blacklist” functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has “whitelist” and “blacklist” functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	With FWTK, in the event of a security policy violation, the user will see their requested web page replaced with an “error” web page, explaining the details of the policy violation. It would be obvious for any other firewall to adopt the same features.
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	<p>My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Under Secure Computing’s proposed construction of a “server that serves as a gateway to the client,” there must be “a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client.” Shaio discloses “SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch.” (3:55-57) Shaio’s Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p>

	<p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no construction, saying that the server is "simply 'an intermediate between the Internet and the user.'" With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio's firewall meets this limitation.</p>
<p>A security policy; An interface for receiving an incoming Downloadable addressed to a client;</p>	<p>Shaio discusses network packets, which include the network address of the destination to which the packets are being transmitted, i.e., the client computer (see, e.g., Shaio 5:1-5). This satisfies "addressed to a client" under both Finjan's and Secure Computing's proposed constructions.</p> <p>Under Secure Computing's proposed construction of a "server that serves as a gateway to the client," there must be "a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client." Shaio discloses "SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch." (3:55-57) Shaio's Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p> <p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no construction, saying that the server is "simply 'an intermediate between the Internet and the user.'" With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio's firewall meets this limitation.</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security</p>	<p>Shaio discloses "additional security may be provided by intelligent firewall... For example, a bytecode verifier may parse the executable code portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses." (5:7-11)</p> <p>The '194 patent explains the concepts of security profile data and suspicious computer operations,</p>

policy has been violated; and	<p>saying "it will be appreciated that the code scanner ... may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker." (5:54-57) Shaio's Java bytecode verifier takes a number of steps, detailed in U.S. patent 5,740,441, incorporated by reference. These steps include, for example, looking at the list of method call operations and field references in the bytecode and verifying that they are properly formed (i.e., they point to a valid class or method which is legal to reference – this is a security policy being enforced by the Java bytecode verifier).</p> <p>In this scenario, the list of bytecodes being verified by Shaio are an example of the Downloadable security profile data in the '194 patent. The rules being evaluated by the bytecode verifier are examples of the security policy in the '194 patent. As a result, Shaio's bytecode verifier meets the limitations of this claim element.</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	Shaio discloses "there is a need for an intelligent firewall that provides real-time security testing of network packets, which may include executable code such as applets, and determines the risk level, i.e., trust worthiness, of each packet before permitting a lower-risk subset of the network packets to execute on anyone of the secure computers" (2:25-31) which meets this limitation of the '194 patent.
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	Shaio discloses "the present invention provides a method and apparatus for determining the trust worthiness of executable packets, e.g., internet applets, being transmitted within a computer network." Shaio satisfies this element of the '194 patent.
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	As above, Shaio discusses "executable packets, e.g., internet applets." ActiveX is another example of an executable packet. Shaio satisfies this element of the '194 patent.
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	As above, Shaio discusses "executable packets, e.g., internet applets." JavaScript scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	As above, Shaio discusses "executable packets, e.g., internet applets." Visual Basic scripts are also examples of executable packets. Shaio satisfies this element of the '194 patent.
65. A computer-readable storage	My analysis of the limitations in claim 65 is the same

<p>medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:</p>	<p>analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Under Secure Computing's proposed construction of a "server that serves as a gateway to the client," there must be "a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client." Shaio discloses "SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch." (3:55-57) Shaio's Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p> <p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no construction, saying that the server is "simply 'an intermediate between the Internet and the user.'" With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio's firewall meets this limitation.</p>
<p>receiving an incoming Downloadable addressed to a client;</p>	<p>Shaio discusses network packets, which include the network address of the destination to which the packets are being transmitted, i.e., the client computer (see, e.g., Shaio 5:1-5). This satisfies "addressed to a client" under both Finjan's and Secure Computing's proposed constructions.</p> <p>Under Secure Computing's proposed construction of a "server that serves as a gateway to the client," there must be "a computer that receives data from its external communications interface and transfers the data through its internal communications interface to the client." Shaio discloses "SWAN ... is coupled to external unsecured computers ... via an externally-accessible network node ... and a public switch." (3:55-57) Shaio's Fig. 1C illustrates that the firewall (185c / 185c1) is coupled both to the external network and to the internal computers.</p> <p>Under Finjan's proposed construction, "a server that serves as a gateway to the client" needs no</p>

	<p>construction, saying that the server is “simply ‘an intermediate between the Internet and the user.’” With this, anything at all, software or hardware, in any location, that sits between a user and the Internet meets their definition. As such, Shaio’s firewall meets this limitation.</p>
<p>comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Shaio discloses “additional security may be provided by intelligent firewall... For example, a bytecode verifier may parse the executable code portion of the packet to eliminate invalid and/or non-conforming instructions in an attempt to reduce the probability of viruses.” (5:7-11)</p> <p>The ‘194 patent explains the concepts of security profile data and suspicious computer operations, saying “it will be appreciated that the code scanner ... may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.” (5:54-57) Shaio’s Java bytecode verifier takes a number of steps, detailed in U.S. patent 5,740,441, incorporated by reference. These steps include, for example, looking at the list of method call operations and field references in the bytecode and verifying that they are properly formed (i.e., they point to a valid class or method which is legal to reference – this is a security policy being enforced by the Java bytecode verifier).</p> <p>In this scenario, the list of bytecodes being verified by Shaio are an example of the Downloadable security profile data in the ‘194 patent. The rules being evaluated by the bytecode verifier are examples of the security policy in the ‘194 patent. As a result, Shaio’s bytecode verifier meets the limitations of this claim element.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated</p>	<p>Shaio discloses “there is a need for an intelligent firewall that provides real-time security testing of network packets, which may include executable code such as applets, and determines the risk level, i.e., trust worthiness, of each packet before permitting a lower-risk subset of the network packets to execute on anyone of the secure computers” (2:25-31) which meets this limitation of the ‘194 patent.</p>

EXHIBIT D

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	FWTK 2.0beta (September 1996)
1. A computer-based method, comprising the steps of:	FWTK, the firewall toolkit, runs on computers.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX).
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	In my analysis, I have been unable to locate a comparison of a list to a security policy having been disclosed in the FWTK source code. I understand that Finjan has accused WebWasher's product of containing a comparison of a list of suspicious computer operations to a security policy. I was unable in my analysis of the WebWasher product to identify a comparison of a list of suspicious computer operations to a security policy. After Finjan identifies a comparison of a list of suspicious computer operations to a security policy in WebWasher's product, my analysis of the FWTK source code may change based on Finjan's interpretation of what constitutes a comparison of a list of suspicious computer operations to a security policy.
preventing execution of the Downloadable by the client if the security policy has been violated.	FWTK optionally removes the suspicious operations (e.g., the applet, script, and object tags) from HTML, thereby preventing the execution of a forbidden Downloadable.
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	FWTK decomposes HTML into its individual elements, which comprise the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further	The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test."

comprising the step of comparing the Downloadable security profile data against the access control list.	(‘194 patent, 8:26-28) FWTK supports a variety of different access control policies (enabling or disabling Java, JavaScript, and/or ActiveX), and will compare HTML elements against these policies. This structure in FWTK is an example of an access control list to which the Downloadable security profile data is compared.
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft’s Authenticode and Sun’s Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a “url-filter” feature, from a configuration file, which satisfies this limitation. FWTK also supports a “-dest” feature, which can specify destination hosts to be allowed or to be denied permission.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK’s “-dest” can specify trusted hosts, to which connections are allowed.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK’s “-dest” can specify untrusted hosts, to which connections are denied.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts (because they use the same <script> tag as is used by JavaScript)
12. The method of claim 1, wherein the security	FWTK’s default policy is applied regardless of the client.

policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host).
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	The host names, specified in the 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	<p>This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art.</p> <p>To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)</p> <p>This limitation describes the behavior of signature-based viruses scanning technologies which were well-known to one of skill in the art. It would be obvious to use such systems as part of an overall firewall system aiming to manage potentially hostile code.</p> <p>Furthermore, one technique for dealing with known hostile code is to "blacklist" code that is known to be non-hostile. FWTK has</p>
25. The method of claim 24, wherein the known Downloadable is hostile.	

	support for whitelisting and blacklisting web sites. Extending this blacklisting support to known hostile code would be obvious to one of skill in the art. To the extent that a reference is needed to disclose signature-based virus scanning technologies, Lo 1994 discloses comparisons between a program and "known malicious code (used by virus scanners)." (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art.
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4) Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27) Hershey and FWTK both disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from both systems together in a single firewall solution.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site.
30. The method of claim 1, further comprising the	In the event of a security policy violation, the user will see their

step of informing a user upon detection of a security policy violation.	requested web page replaced with an "error" web page, explaining the details of the policy violation.
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.
A security policy;	FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX).
An interface for receiving an incoming Downloadable addressed to a client;	FWTK optionally removes the suspicious operations (e.g., the applet, script, and object tags) from HTML, thereby preventing the execution of a forbidden Downloadable
A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX). In my analysis, I have been unable to locate a comparison of a list to a security policy having been disclosed in the FWTK source code. I understand that Finjan has accused WebWasher's product of containing a comparison of a list of suspicious computer operations to a security policy. I was unable in my analysis of the WebWasher product to identify a comparison of a list of suspicious computer operations to a security policy. After Finjan identifies a comparison of a list of suspicious computer operations to a security policy in WebWasher's product, my analysis of the FWTK source code may change based on Finjan's interpretation of what constitutes a comparison of a list of suspicious computer operations to a security policy.
A logical engine for preventing execution by the Downloadable by the client if the security policy has	FWTK optionally removes the suspicious operations (e.g., the applet, script, and object tags) from HTML, thereby preventing the

been violated.	execution of a forbidden Downloadable
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets as described in my analysis of claim 8 above.
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls as described in my analysis of claim 9 above.
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts as described in my analysis of claim 10 above.
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts (because they use the same <script> tag as is used by JavaScript), as described in my analysis of claim 11 above.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below. FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX).
receiving an incoming Downloadable addressed to a client;	FWTK acts as gateway (an application-level proxy server), running on a server, through which a client will access the Internet and through which a client will receive Downloadables (e.g., Java, JavaScript, and/or ActiveX).
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	In my analysis, I have been unable to locate a comparison of a list to a security policy having been disclosed in the FWTK source code. I understand that Finjan has accused WebWasher's product of containing a comparison of a list of suspicious computer operations to a security policy. I was unable in my analysis of the WebWasher product to identify a comparison of a list of suspicious computer operations to a security policy. After Finjan identifies a comparison of a list of suspicious computer operations to a security

	policy in WebWasher's product, my analysis of the FWTK source code may change based on Finjan's interpretation of what constitutes a comparison of a list of suspicious computer operations to a security policy.
preventing execution of the Downloadable by the client if the security policy has been violated	FWTK optionally removes the suspicious operations (e.g., the applet, script, and object tags) from HTML, thereby preventing the execution of a forbidden Downloadable

EXHIBIT E

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Hershey (U.S. Patent 5,414,833) and Lo et al. (1991)
1. A computer-based method, comprising the steps of:	Both Hershey and Lo disclose computer-based methods.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	Lo discloses techniques for performing "malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.
	Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code. Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of

	<p>these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent references this specific article (5:3-10). Consequently, one of ordinary skill in the art would have been specifically motivated to combine it with Hershey.</p> <p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>Lo discloses decomposing a compiled program into its assembly-language form ("The first module, the disassembler, takes an executable program as input and produces the equivalent Motorola 68020 assembly language representation as out.", p. 163). This disassembled program comprises the Downloadable security profile data.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	

3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Lo discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163). This comprises a list that includes the criteria necessary for the Downloadable to fail or pass the test.</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to

	<p>one of skill in the art and would be obviously applicable to any other firewall.</p> <p>To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4)</p> <p>Hershey discloses a copending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.</p> <p>FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.</p> <p>Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (19:40-41)</p> <p>My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a</p>
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	
A security policy;	

	<p>security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).</p>
<p>An interface for receiving an incoming Downloadable addressed to a client;</p>	<p>Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing "malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus</p>

	<p>detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent references this specific article (5:3-10). Consequently, one of ordinary skill in the art would have been specifically motivated to combine it with Hershey.</p> <p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out Visual Basic scripts. FWTK was a widely</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	
36. The system of claim 32, wherein the	

Downloadable includes a Visual Basic script.	known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.
receiving an incoming Downloadable addressed to a client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	<p>Lo discloses techniques for performing "malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of</p>

	<p>these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent references this specific article (5:3-10). Consequently, one of ordinary skill in the art would have been specifically motivated to combine it with Hershey.</p>
preventing execution of the Downloadable by the client if the security policy has been violated	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p>

EXHIBIT F

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Hershey (U.S. Patent 5,414,833) and Lo et al. (1994)
1. A computer-based method, comprising the steps of:	Both Hershey and Lo disclose computer-based methods.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.
	Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code. Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these

	<p>against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent specifically references previous work, completed by the same author, related to static analysis of malicious code. Consequently, one of ordinary skill in the art would have been specifically motivated to discover other work by Lo and to combine it with Hershey.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the	<p>Lo discloses a variety of "tell-tale signs" which can be determined from all kinds of programs by using its "program slicer". These</p>

Downloadable security profile data.	tell-tale signs include File Read, File Write, Program Execution, Network Access, and a variety of other items (p. 5-6). The process of determining these tell-tale signs is an example of decomposing the Downloadable into the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Lo discloses "Some program properties allow us to discern malicious programs from benign programs easily with very high accuracy without the need to give a specification of the program. We call those properties <i>tell-tale</i> signs. The idea is to program a <i>filter</i> to identify these tell-tale signs." (p. 3) Lo describes a list of these tell-tale signs (p. 5-6). This structure in Lo is an example of an access control list to which the Downloadable security profile data is compared.</p> <p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p> <p>FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	
6. The method of claim 5, wherein the known URL is a trusted URL.	

	these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)

25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	Lo discloses comparisons between a program and "a 'clean' copy of the program" (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a compending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (19:40-41)
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below. Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
A security policy;	Lo discloses Downloadable security profile data ("The idea is to

	<p>program a <i>filter</i> to identify these tell-tale signs.”, p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the “tell-tale signs”, p. 5-6, including “file read,” “file write,” “process creation,” “program execution,” “network accesses,” and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: “The files written to should be checked against a list of important system files.”, p. 5).</p>
An interface for receiving an incoming Downloadable addressed to a client;	<p>Hershey discloses “real-time detection of the presence of a suspected offending virus” (abstract) and describes performing this on a network gateway (“Security Agent (SA)”, Figs. 3 and 13).</p>
A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	<p>Lo discloses techniques for performing “mechanized malicious code detection.” Under Secure Computing’s proposed construction for “Downloadable” (“a program or document containing an executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that “code” is executable. As such, Lo’s “code” is an example of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data (“The idea is to program a <i>filter</i> to identify these tell-tale signs.”, p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the “tell-tale signs”, p. 5-6, including “file read,” “file write,” “process creation,” “program execution,” “network accesses,” and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for</p>

	<p>example: "The files written to should be checked against a list of important system files," p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent specifically references previous work, completed by the same author, related to static analysis of malicious code. Consequently, one of ordinary skill in the art would have been specifically motivated to discover other work by Lo and to combine it with Hershey.</p> <p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	

34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.
receiving an incoming Downloadable addressed to a client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.
	Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.

	<p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (5:7-9)</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Hershey patent specifically references previous work, completed by the same author, related to static analysis of malicious code. Consequently, one of ordinary skill in the art would have been specifically motivated to discover other work by Lo and to combine it with Hershey.</p> <p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (17:1-5)</p>
preventing execution of the Downloadable by the client if the security policy has been violated	

	By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.
--	--

EXHIBIT G

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Hershey (U.S. Patent 5,414,833) and Chen (U.S. Patent 5,951,698)
1. A computer-based method, comprising the steps of:	Both Hershey and Chen disclose computer-based methods.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and	Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17) Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent. Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent. Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of

ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.

Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.

Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).

It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (Hershey 4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (Hershey 5:7-9)

Chen was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Chen patent specifically references Hershey. Consequently, one of ordinary skill in the art would have been specifically motivated to combine Chen and Hershey.

preventing execution of the Downloadable by the client if the security policy has been violated.	<p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (Hershey 17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>Chen also discloses techniques to prevent execution: "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file." (Chen 17:21-25)</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	<p>Chen discloses the decomposition of a macro: "once a set of instruction identifiers is obtained by the macro virus scanning module ... the decoded macro is scanned..." (15:14-16). The "set of instruction identifiers" comprises the Downloadable security profile data.</p>
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Chen discloses the comparison of the decoded macro (an example of Downloadable security profile data) to an access control list, an example of which is provided in Fig. 9. See also: "The comparison data in the virus information module preferably includes several sets of instruction identifiers. Various combinations of suspect instructions may be detected using the sets of instruction identifiers. Various different macro virus enablement and/or macro virus reproduction instructions may be identified using each set of</p>

	instruction identifiers. The instruction identifiers are not restricted to macro virus enablement and reproduction instructions. For example, an instruction which causes the computer hard disk to be reformatted without verification and an instruction which changes the system settings to allow such reformatting without user notice could be used as a suspect instruction combination." (14:52-64)
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code. Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the	FWTK can filter out ActiveX controls. FWTK was a widely

Downloadable includes an ActiveX™ control.	known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Hershey and Chen apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
25. The method of claim 24, wherein the known Downloadable is hostile.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obviously applicable to any other firewall.
	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program

	and "a 'clean' copy of the program." (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a copending application which "provides methods and apparatus to automatically detect and extract signature from an undesirable software entity ... It further provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:21-27) This describes the inclusion of a previously received Downloadable as a known Downloadable.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	Hershey discloses "a virus alert message can be broadcast to system users if a virus is detected." (Hershey 19:40-41)
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	Chen discloses "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus." (Chen 17:21-23)
	My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.
A security policy;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13). Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the

<p>An interface for receiving an incoming Downloadable addressed to a client;</p>	<p>comparison "against a security policy" in the '194 patent. Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p>

	<p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that "it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks." (Hershey 4:54-58) Hershey then describes existing techniques, including static analysis tools, "which have been shown to be effective against certain types of malicious code." (Hershey 5:7-9)</p> <p>Chen was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Chen patent specifically references Hershey. Consequently, one of ordinary skill in the art would have been specifically motivated to combine Chen and Hershey.</p> <p>Hershey discloses "adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways." (Hershey 17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	

	discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated. Chen also discloses techniques to prevent execution: "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file." (Chen 17:21-25)
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.
receiving an incoming Downloadable addressed to a client;	Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13). Hershey discloses "real-time detection of the presence of a suspected offending virus" (abstract) and describes performing this on a network gateway ("Security Agent (SA)", Figs. 3 and 13).
comparing Downloadable security profile data	Chen discloses "once a set of instruction identifiers is obtained by

<p>pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers.” (Chen 15:13-17)</p> <p>Chen’s “set of instruction identifiers” is an example of the “list of suspicious computer operations” in the ‘194 patent.</p> <p>Chen’s determination whether those identifiers include “a combination of suspect instructions” is an example of the comparison “against a security policy” in the ‘194 patent.</p> <p>Under Secure Computing’s proposed construction for “Downloadable” (“a program or document containing an executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that Chen’s macros are executable. As such, Chen’s “macros” are examples of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the “decoded macro”). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including “MacroCopy,” “FileSaveAs”, discussed further in column 14). Chen discloses comparing these against a security policy (“the decoded macro is scanned to determine whether it includes a combination of suspect instructions</p>
--	--

	<p>as identified by the instruction identifiers”, Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Hershey. In particular, Hershey itself discloses that “it has become clear to many people in the industry that methods for automatically recognizing and eradicating previously unknown or unanalyzed viruses must be developed and installed on individual computers and computer networks.” (Hershey 4:54-58) Hershey then describes existing techniques, including static analysis tools, “which have been shown to be effective against certain types of malicious code.” (Hershey 5:7-9)</p> <p>Chen was a well-known static analysis technique that was effective against certain types of malicious code. Moreover, the Chen patent specifically references Hershey. Consequently, one of ordinary skill in the art would have been specifically motivated to combine Chen and Hershey.</p> <p>Hershey discloses “adaptive, active monitoring means ... capable of detecting a plurality of characteristic patterns, in which case responder ... is capable of modifying, injecting, and deleting information in bit stream ... in a plurality of ways.” (Hershey 17:1-5)</p> <p>By modifying or deleting information in the bit stream, Hershey discloses the prevention of execution of a Downloadable by the client. Likewise, by detecting a plurality of characteristic patterns, Hershey determines if the security policy has been violated.</p> <p>Chen also discloses techniques to prevent execution: “Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected</p>
preventing execution of the Downloadable by the client if the security policy has been violated	

	macro from the targeted file without replacement, or deleting the targeted file." (Chen 17:21-25)
--	---

EXHIBIT 2, PART 2

EXHIBIT H

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Lo et al. (1991)
1. A computer-based method, comprising the steps of:	Both Ji and Lo disclose computer-based methods.
receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing "malicious code detection."</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this</p>

	<p>construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p>
<p>preventing execution of the Downloadable by the client if the security policy has been violated.</p>	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p>

	By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	Lo discloses decomposing a compiled program into its assembly-language form ("The first module, the disassembler, takes an executable program as input and produces the equivalent Motorola 68020 assembly language representation as out.", p. 163). This disassembled program comprises the Downloadable security profile data.
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The '194 patent's "access control list" is a "list that includes the criteria necessary for the Downloadable to fail or pass the test." ('194 patent, 8:26-28)</p> <p>Lo discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163). This comprises a list that includes the criteria necessary for the Downloadable to fail or pass the test.</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other

	firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK's "dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK's "dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Ji and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses "a scanner will search a program for patterns which match those of known malicious programs." (p. 162)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obviously applicable to any other firewall.
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4) Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27) Hershey, Ji, and Lo disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each systems together in a single firewall solution. FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features. FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features. In the context of a FTP file transfer, when a virus is detected, Ji
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	
30. The method of claim 1, further comprising the	

step of informing a user upon detection of a security policy violation.	discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below. Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44) Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)
A security policy;	Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).
An interface for receiving an incoming Downloadable addressed to a client;	Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes

	<p>and a gateway node for connection to other networks.” (2:42-44)</p> <p>Ji discloses a variety of applications: “While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web.” (5:28-35)</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing “malicious code detection.” Under Secure Computing’s proposed construction for “Downloadable” (“a program or document containing an executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that “code” is executable. As such, Lo’s “code” is an example of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Lo discloses a list of suspicious computer operations that may be attempted (“The second module, the analyzer, takes the output from the disassembler and examines it . . .”, p. 163) and discloses a security policy (“examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls”, p. 163). Lo gives examples of these system calls (“e.g., open, write, close”, p. 163).</p> <p>It would have been obvious for one of ordinary skill in the art in</p>

	<p>1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p> <p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	

<p>36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.</p>	<p>obvious for any other firewall to adopt the same features. FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
<p>65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:</p>	<p>My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>receiving an incoming Downloadable addressed to a client;</p>	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>

comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and

Lo discloses techniques for performing "malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.

Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.

Lo discloses a list of suspicious computer operations that may be attempted ("The second module, the analyzer, takes the output from the disassembler and examines it ...", p. 163) and discloses a security policy ("examines it for duplication of OS services, reporting any such duplications as well as the number of occurrences of all system calls", p. 163). Lo gives examples of these system calls ("e.g., open, write, close", p. 163).

It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).

Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.

preventing execution of the Downloadable by the client if the security policy has been violated

Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)

Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)

By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.

EXHIBIT I

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Lo et al. (1994)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Lo disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this</p>

	<p>construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to</p>

	<p>a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus.” (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Lo discloses a variety of “tell-tale signs” which can be determined from all kinds of programs by using its “program slicer”. These tell-tale signs include File Read, File Write, Program Execution, Network Access, and a variety of other items (p. 5-6). The process of determining these tell-tale signs is an example of decomposing the Downloadable into the Downloadable security profile data.</p> <p>The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Lo discloses “Some program properties allow us to discern malicious programs from benign programs easily with very high accuracy without the need to give a specification of the program. We call those properties <i>tell-tale</i> signs. The idea is to program a <i>filter</i> to identify these tell-tale signs.” (p. 3) Lo describes a list of these tell-tale signs (p. 5-6). This structure in Lo is an example of an access control list to which the Downloadable security profile data is compared.</p> <p>Microsoft’s Authenticode and Sun’s Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p> <p>FWTK supports a “url-filter” feature, from a configuration file,</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	
5. The method of claim 1, further comprising the	

step of comparing the URL from which the Downloadable originated against a known URL.	which satisfies this limitation. FWTK also supports a “-dest” feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
6. The method of claim 5, wherein the known URL is a trusted URL.	FWTK’s “-dest” can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
7. The method of claim 5, wherein the known URL is an untrusted URL.	FWTK’s “-dest” can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	FWTK can filter out Visual Basic scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Ji and Lo apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the ‘hosts’ configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.

14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
25. The method of claim 24, wherein the known Downloadable is hostile.	Lo discloses comparisons between a program and "known malicious code (used by virus scanners)" (p. 4)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	Lo discloses comparisons between a program and "a 'clean' copy of the program" (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27)
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	Hershey, Ji, and Lo disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each systems together in a single firewall solution.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features. In the context of a FTP file transfer, when a virus is detected, Ji discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)

<p>32. A system for execution by a server that serves as a gateway to a client, the system comprising:</p>	<p>My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>A security policy;</p>	<p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p>
<p>An interface for receiving an incoming Downloadable addressed to a client;</p>	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and</p>

	<p>preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that "code" is executable. As such, Lo's "code" is an example of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data ("The idea is to program a <i>filter</i> to identify these tell-tale signs.", p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the "tell-tale signs", p. 5-6, including "file read," "file write," "process creation," "program execution," "network accesses," and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: "The files written to should be checked against a list of important system files.", p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji</p>

	<p>itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p> <p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out Visual Basic scripts. FWTK was a widely</p>
A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.	
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	
36. The system of claim 32, wherein the	

Downloadable includes a Visual Basic script.	known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	<p>My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
receiving an incoming Downloadable addressed to a client;	Lo discloses techniques for performing "mechanized malicious code detection." Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious	

computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	<p>executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that “code” is executable. As such, Lo’s “code” is an example of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Lo discloses Downloadable security profile data (“The idea is to program a <i>filter</i> to identify these tell-tale signs,” p. 3). Lo discloses suspicious computer operations that may be attempted by the Downloadable (the “tell-tale signs”, p. 5-6, including “file read,” “file write,” “process creation,” “program execution,” “network accesses,” and so forth). Lo discloses comparing these against a security policy (described, by examples, on p. 5-6, for example: “The files written to should be checked against a list of important system files,” p. 5).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Lo with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that “those skilled in the art will realize that various other virus detection methods may also be used” (7:63-65).</p> <p>Lo was a well-known static analysis technique that was effective against certain types of malicious code. Consequently, one of ordinary skill in the art would have been motivated to combine it with Ji.</p> <p>Ji discloses “the temporarily stored file is analyzed to determine if it</p>
preventing execution of the Downloadable by the	

client if the security policy has been violated	<p>contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p>
---	---

EXHIBIT 2, PART 3

EXHIBIT J

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Chen (U.S. Patent 5,951,698)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Chen disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p>

	<p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves

	<p>the configuration file to determine the handling of the temporary file.” (9:6-11)</p> <p>Ji also discloses “a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus.” (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: “Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	<p>Chen discloses the decomposition of a macro: “once a set of instruction identifiers is obtained by the macro virus scanning module ... the decoded macro is scanned...” (15:14-16). The “set of instruction identifiers” comprises the Downloadable security profile data.</p>
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Chen discloses the comparison of the decoded macro (an example of Downloadable security profile data) to an access control list, an example of which is provided in Fig. 9. See also: “The comparison data in the virus information module preferably includes several sets of instruction identifiers. Various combinations of suspect</p>

	<p>instructions may be detected using the sets of instruction identifiers. Various different macro virus enablement and/or macro virus reproduction instructions may be identified using each set of instruction identifiers. The instruction identifiers are not restricted to macro virus enablement and reproduction instructions. For example, an instruction which causes the computer hard disk to be reformatted without verification and an instruction which changes the system settings to allow such reformatting without user notice could be used as a suspect instruction combination." (14:52-64)</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	<p>FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "-dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
6. The method of claim 5, wherein the known URL is a trusted URL.	<p>FWTK's "-dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
7. The method of claim 5, wherein the known URL is an untrusted URL.	<p>FWTK's "-dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Ji and Chen apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
25. The method of claim 24, wherein the known Downloadable is hostile.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obvious to apply to any firewall

	system. To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27) Hershey, Ji, and Chen disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each system together in a single firewall solution.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	In the context of a FTP file transfer, when a virus is detected, Ji discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	Chen discloses "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus." (Chen 17:21-23) My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have

	<p>previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p>
A security policy;	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP</p>
An interface for receiving an incoming Downloadable addressed to a client;	

	<p>proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web.” (5:28-35)</p> <p>Chen discloses “once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers.” (Chen 15:13-17)</p> <p>Chen’s “set of instruction identifiers” is an example of the “list of suspicious computer operations” in the ‘194 patent.</p> <p>Chen’s determination whether those identifiers include “a combination of suspect instructions” is an example of the comparison “against a security policy” in the ‘194 patent.</p> <p>Under Secure Computing’s proposed construction for “Downloadable” (“a program or document containing an executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that Chen’s macros are executable. As such, Chen’s “macros” are examples of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the “decoded</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	

	<p>macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
<p>A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.</p>	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: "Various</p>

	alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below. Ji discloses a virus detection on a gateway, e.g., a “system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks.” (2:42-44) Ji discloses a variety of applications: “While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web.” (5:28-35)

receiving an incoming Downloadable addressed to a client;	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p>
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	

	<p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
preventing execution of the Downloadable by the client if the security policy has been violated	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the</p>

	<p>file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file." (Chen 17:21-25)</p>
--	--

EXHIBIT 2, PART 3

EXHIBIT J

FINJAN SOFTWARE v. SECURE COMPUTING

Charts for 6,092,194

Claim Language '194 Patent	Ji (U.S. Patent 5,623,600) and Chen (U.S. Patent 5,951,698)
<p>1. A computer-based method, comprising the steps of:</p> <p>receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;</p>	<p>Both Ji and Chen disclose computer-based methods.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p>
<p>comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list of suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p>

	<p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
preventing execution of the Downloadable by the client if the security policy has been violated.	Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves

	<p>the configuration file to determine the handling of the temporary file.” (9:6-11)</p> <p>Ji also discloses “a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus.” (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: “Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)</p>
2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.	<p>Chen discloses the decomposition of a macro: “once a set of instruction identifiers is obtained by the macro virus scanning module ... the decoded macro is scanned...” (15:14-16). The “set of instruction identifiers” comprises the Downloadable security profile data.</p>
3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.	<p>The ‘194 patent’s “access control list” is a “list that includes the criteria necessary for the Downloadable to fail or pass the test.” (‘194 patent, 8:26-28)</p> <p>Chen discloses the comparison of the decoded macro (an example of Downloadable security profile data) to an access control list, an example of which is provided in Fig. 9. See also: “The comparison data in the virus information module preferably includes several sets of instruction identifiers. Various combinations of suspect</p>

	<p>instructions may be detected using the sets of instruction identifiers. Various different macro virus enablement and/or macro virus reproduction instructions may be identified using each set of instruction identifiers. The instruction identifiers are not restricted to macro virus enablement and reproduction instructions. For example, an instruction which causes the computer hard disk to be reformatted without verification and an instruction which changes the system settings to allow such reformatting without user notice could be used as a suspect instruction combination." (14:52-64)</p>
4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.	<p>Microsoft's Authenticode and Sun's Signed Java both support the use of digital signatures and their associated cryptographic certificates, to sign code.</p> <p>Because the purpose of scanning certificates is to determine whether or not outside code carries an endorsement which implies we should trust it, scanning these certificates would be an obvious part of an overall firewall system aiming to manage potentially hostile code.</p>
5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.	<p>FWTK supports a "url-filter" feature, from a configuration file, which satisfies this limitation. FWTK also supports a "dest" feature, which can specify destination hosts to be allowed or to be denied permission. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
6. The method of claim 5, wherein the known URL is a trusted URL.	<p>FWTK's "dest" can specify trusted hosts, to which connections are allowed. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>
7. The method of claim 5, wherein the known URL is an untrusted URL.	<p>FWTK's "dest" can specify untrusted hosts, to which connections are denied. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p>

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.	FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.	FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.	FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.
11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.	The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.
12. The method of claim 1, wherein the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.	Ji and Chen apply their security policy regardless of the client to whom the Downloadable is addressed.
13. The method of claim 1, wherein the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.	FWTK supports host-specific policies (the 'hosts' configuration item can specify a list of hosts and a different policy for each host). It would be obvious for any other firewall to adopt the same features.
14. The method of claim 1, wherein the client belongs to a particular group; and the security policy includes a specific security policy corresponding to the particular group.	FWTK's host names, specified in its 'hosts' configuration, may specify "wildcard" patterns that match groups of hosts, and then each group may have its own policy. It would be obvious for any other firewall to adopt the same features.
24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
25. The method of claim 24, wherein the known Downloadable is hostile.	Chen discloses "First, a decoded macro is scanned for known viruses." (2:53-54)
26. The method of claim 24, wherein the known Downloadable is non-hostile.	One technique for dealing with known non-hostile code is to "whitelist" code that is known to be non-hostile. FWTK has support for whitelisting and blacklisting web sites. Extending this whitelisting support to known non-hostile code would be obvious to one of skill in the art and would be obvious to apply to any firewall

	system. To the extent that a reference is needed to disclose allowing non-hostile code, Lo 1994 discloses comparisons between a program and "a 'clean' copy of the program." (p. 4)
27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.	Hershey discloses a computer system that "provides methods and apparatus for immunizing a computer system ... against a subsequent infection by a previously unknown and undesirable software entity." (5:23-27) Hershey, Ji, and Chen disclose different techniques for providing protection against potentially hostile code. It would have been obvious for one of ordinary skill in the art to use techniques from each system together in a single firewall solution.
28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to block code from a particular web site. It would be obvious for any other firewall to adopt the same features.
29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.	FWTK has "whitelist" and "blacklist" functionality that can be applied to arbitrary URLs. This can be used to implement an administrative override to allow code from a particular web site. It would be obvious for any other firewall to adopt the same features.
30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.	In the context of a FTP file transfer, when a virus is detected, Ji discloses, in one embodiment, that "the file is renamed and stored in a specified directory on the gateway node and the user is notified of the new file name and directory path which can be used to manually request the file from the system administrator." (8:28-31)
32. A system for execution by a server that serves as a gateway to a client, the system comprising:	Chen discloses "Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus." (Chen 17:21-23) My analysis of the limitations in claim 32 is the same analysis as claim 1. In order to explicitly align the comparisons that I have

	<p>previously made for claim 1 with the limitations of claim 32, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p>
A security policy;	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP</p>
An interface for receiving an incoming Downloadable addressed to a client;	

	<p>proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web.” (5:28-35)</p>
<p>A comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and</p>	<p>Chen discloses “once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers.” (Chen 15:13-17)</p> <p>Chen’s “set of instruction identifiers” is an example of the “list of suspicious computer operations” in the ‘194 patent.</p> <p>Chen’s determination whether those identifiers include “a combination of suspect instructions” is an example of the comparison “against a security policy” in the ‘194 patent.</p> <p>Under Secure Computing’s proposed construction for “Downloadable” (“a program or document containing an executable application program that can be downloaded from one computer to another computer”), one of ordinary skill in the art would understand that Chen’s macros are executable. As such, Chen’s “macros” are examples of a “program or document containing an executable application program.” It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p> <p>Under Finjan’s proposed construction for “Downloadable” (“program or document containing mobile code”). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the “decoded</p>

	<p>macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
<p>A logical engine for preventing execution by the Downloadable by the client if the security policy has been violated.</p>	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the file if the file contains a virus." (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: "Various</p>

	<p>alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)</p> <p>FWTK can filter out Java applets. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out ActiveX controls. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>FWTK can filter out JavaScript scripts. FWTK was a widely known and available firewall with these features. It would be obvious for any other firewall to adopt the same features.</p> <p>The macros discussed by Chen refer to Microsoft Word (see, e.g., 5:22 or 13:64). Microsoft Word uses Visual Basic for its macros.</p> <p>My analysis of the limitations in claim 65 is the same analysis as claim 1. In order to explicitly align the comparisons that I have previously made for claim 1 with the limitations of claim 65, I have placed the appropriate text from my analysis of claim 1 next to the limitations below.</p> <p>Ji discloses a virus detection on a gateway, e.g., a “system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks.” (2:42-44)</p> <p>Ji discloses a variety of applications: “While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web.” (5:28-35)</p>
33. The system of claim 32, wherein the Downloadable includes a Java™ applet.	
34. The system of claim 32, wherein the Downloadable includes an ActiveX™ control.	
35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.	
36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.	
65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:	

receiving an incoming Downloadable addressed to a client;	<p>Ji discloses a virus detection on a gateway, e.g., a "system including the present invention is a network formed of a plurality of nodes and a gateway node for connection to other networks." (2:42-44)</p> <p>Ji discloses a variety of applications: "While the apparatus of the present invention, in particular the FTP proxy server and SMTP proxy server, has been described above as being located and preferably is located on the gateway node, those skilled in the art will realize that the apparatus of the present invention could also be included on a FTP server or a world wide web server for scanning file and messages as they are downloaded from the web." (5:28-35)</p> <p>Chen discloses "once a set of instruction identifiers is obtained by the macro virus scanning module..., the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers." (Chen 15:13-17)</p> <p>Chen's "set of instruction identifiers" is an example of the "list of suspicious computer operations" in the '194 patent.</p> <p>Chen's determination whether those identifiers include "a combination of suspect instructions" is an example of the comparison "against a security policy" in the '194 patent.</p> <p>Under Secure Computing's proposed construction for "Downloadable" ("a program or document containing an executable application program that can be downloaded from one computer to another computer"), one of ordinary skill in the art would understand that Chen's macros are executable. As such, Chen's "macros" are examples of a "program or document containing an executable application program." It is well known by one of ordinary skill in the art that such programs or documents can be downloaded from one computer to another computer.</p>
comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and	

	<p>Under Finjan's proposed construction for "Downloadable" ("program or document containing mobile code"). Under this construction, macros are mobile code.</p> <p>Chen discloses Downloadable security profile data (the "decoded macro"). Chen discloses suspicious computer operations that may be attempted by the Downloadable (Fig. 9 lists several exemplary suspicious operations, including "MacroCopy," "FileSaveAs", discussed further in column 14). Chen discloses comparing these against a security policy ("the decoded macro is scanned to determine whether it includes a combination of suspect instructions as identified by the instruction identifiers", Chen 15:15-19).</p> <p>It would have been obvious for one of ordinary skill in the art in 1996 to combine the techniques disclosed in Chen with the virus detection devices and techniques disclosed in Ji. In particular, Ji itself discloses that "those skilled in the art will realize that various other virus detection methods may also be used" (7:63-65).</p>
preventing execution of the Downloadable by the client if the security policy has been violated	<p>Ji discloses "the temporarily stored file is analyzed to determine if it contains viruses... If no viruses are detected, the method continues ... However, if a virus is detected, the present invention retrieves the configuration file to determine the handling of the temporary file." (9:6-11)</p> <p>Ji also discloses "a method for processing a file before transmission into or from the network includes the steps of: receiving the data transfer command and file name; transferring the file to a system node; performing virus detection on the file; determining whether the file contains any viruses; transferring the file from the system to a recipient node if the file does not contain a virus; and deleting the</p>

	<p>file if the file contains a virus.” (Abstract)</p> <p>By deleting files with viruses rather than transferring them to the client, Ji prevents their execution.</p> <p>Chen also discloses techniques to prevent execution: “Various alternative corrective steps will be recognized such as notifying the user that the targeted file includes a virus, removing the infected macro from the targeted file without replacement, or deleting the targeted file.” (Chen 17:21-25)</p>
--	--

EXHIBIT 3

**THIS EXHIBIT HAS BEEN
REDACTED IN ITS ENTIRETY**